



Stellaris® LM3S9L97 Microcontroller

DATA SHEET

Copyright

Copyright © 2007-2010 Texas Instruments Incorporated All rights reserved. Stellaris and StellarisWare are registered trademarks of Texas Instruments Incorporated. ARM and Thumb are registered trademarks and Cortex is a trademark of ARM Limited. Other names and brands may be claimed as the property of others.

ADVANCE INFORMATION concerns new products in the sampling or preproduction phase of development. Characteristic data and other specifications are subject to change without notice.

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

Texas Instruments Incorporated

108 Wild Basin, Suite 350

Austin, TX 78746

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

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



Table of Contents

Revision History	39
About This Document	46
Audience	46
About This Manual	46
Related Documents	46
Documentation Conventions	47
1 Architectural Overview	49
1.1 Functional Overview	51
1.1.1 ARM Cortex™-M3	51
1.1.2 On-Chip Memory	53
1.1.3 Serial Communications Peripherals	54
1.1.4 System Integration	60
1.1.5 Advanced Motion Control	66
1.1.6 Analog	68
1.1.7 JTAG and ARM Serial Wire Debug	69
1.1.8 Packaging and Temperature	70
1.2 Target Applications	70
1.3 High-Level Block Diagram	70
1.4 Hardware Details	72
2 The Cortex-M3 Processor	73
2.1 Block Diagram	74
2.2 Overview	75
2.2.1 System-Level Interface	75
2.2.2 Integrated Configurable Debug	75
2.2.3 Trace Port Interface Unit (TPIU)	76
2.2.4 Cortex-M3 System Component Details	76
2.3 Programming Model	77
2.3.1 Processor Mode and Privilege Levels for Software Execution	77
2.3.2 Stacks	77
2.3.3 Register Map	78
2.3.4 Register Descriptions	79
2.3.5 Exceptions and Interrupts	92
2.3.6 Data Types	92
2.4 Memory Model	92
2.4.1 Memory Regions, Types and Attributes	94
2.4.2 Memory System Ordering of Memory Accesses	95
2.4.3 Behavior of Memory Accesses	95
2.4.4 Software Ordering of Memory Accesses	96
2.4.5 Bit-Banding	97
2.4.6 Data Storage	99
2.4.7 Synchronization Primitives	100
2.5 Exception Model	101
2.5.1 Exception States	101
2.5.2 Exception Types	102
2.5.3 Exception Handlers	105

2.5.4	Vector Table	105
2.5.5	Exception Priorities	106
2.5.6	Interrupt Priority Grouping	107
2.5.7	Exception Entry and Return	107
2.6	Fault Handling	109
2.6.1	Fault Types	109
2.6.2	Fault Escalation and Hard Faults	110
2.6.3	Fault Status Registers and Fault Address Registers	111
2.6.4	Lockup	111
2.7	Power Management	111
2.7.1	Entering Sleep Modes	111
2.7.2	Wake Up from Sleep Mode	112
2.8	Instruction Set Summary	112
3	Cortex-M3 Peripherals	116
3.1	Functional Description	116
3.1.1	System Timer (SysTick)	116
3.1.2	Nested Vectored Interrupt Controller (NVIC)	117
3.1.3	System Control Block (SCB)	119
3.1.4	Memory Protection Unit (MPU)	119
3.2	Register Map	124
3.3	System Timer (SysTick) Register Descriptions	126
3.4	NVIC Register Descriptions	130
3.5	System Control Block (SCB) Register Descriptions	143
3.6	Memory Protection Unit (MPU) Register Descriptions	172
4	JTAG Interface	182
4.1	Block Diagram	183
4.2	Signal Description	183
4.3	Functional Description	184
4.3.1	JTAG Interface Pins	184
4.3.2	JTAG TAP Controller	186
4.3.3	Shift Registers	186
4.3.4	Operational Considerations	187
4.4	Initialization and Configuration	189
4.5	Register Descriptions	190
4.5.1	Instruction Register (IR)	190
4.5.2	Data Registers	192
5	System Control	194
5.1	Signal Description	194
5.2	Functional Description	194
5.2.1	Device Identification	195
5.2.2	Reset Control	195
5.2.3	Non-Maskable Interrupt	202
5.2.4	Power Control	203
5.2.5	Clock Control	203
5.2.6	System Control	211
5.3	Initialization and Configuration	212
5.4	Register Map	212
5.5	Register Descriptions	214

6	Hibernation Module	301
6.1	Block Diagram	302
6.2	Signal Description	302
6.3	Functional Description	303
6.3.1	Register Access Timing	304
6.3.2	Hibernation Clock Source	304
6.3.3	Battery Management	306
6.3.4	Real-Time Clock	306
6.3.5	Non-Volatile Memory	307
6.3.6	Power Control Using HIB	307
6.3.7	Power Control Using VDD3ON Mode	307
6.3.8	Initiating Hibernate	307
6.3.9	Interrupts and Status	308
6.4	Initialization and Configuration	308
6.4.1	Initialization	308
6.4.2	RTC Match Functionality (No Hibernation)	309
6.4.3	RTC Match/Wake-Up from Hibernation	309
6.4.4	External Wake-Up from Hibernation	310
6.4.5	RTC or External Wake-Up from Hibernation	310
6.4.6	Register Reset	310
6.5	Register Map	311
6.6	Register Descriptions	311
7	Internal Memory	328
7.1	Block Diagram	328
7.2	Functional Description	328
7.2.1	SRAM	329
7.2.2	ROM	329
7.2.3	Flash Memory	331
7.3	Flash Memory Initialization and Configuration	333
7.3.1	Flash Memory Programming	333
7.3.2	32-Word Flash Memory Write Buffer	334
7.3.3	Nonvolatile Register Programming	335
7.4	Register Map	336
7.5	Flash Memory Register Descriptions (Flash Control Offset)	337
7.6	Memory Register Descriptions (System Control Offset)	349
8	Micro Direct Memory Access (μDMA)	365
8.1	Block Diagram	366
8.2	Functional Description	366
8.2.1	Channel Assignments	367
8.2.2	Priority	368
8.2.3	Arbitration Size	368
8.2.4	Request Types	368
8.2.5	Channel Configuration	369
8.2.6	Transfer Modes	371
8.2.7	Transfer Size and Increment	379
8.2.8	Peripheral Interface	379
8.2.9	Software Request	379
8.2.10	Interrupts and Errors	380

8.3	Initialization and Configuration	380
8.3.1	Module Initialization	380
8.3.2	Configuring a Memory-to-Memory Transfer	380
8.3.3	Configuring a Peripheral for Simple Transmit	382
8.3.4	Configuring a Peripheral for Ping-Pong Receive	383
8.3.5	Configuring Channel Assignments	386
8.4	Register Map	386
8.5	μDMA Channel Control Structure	387
8.6	μDMA Register Descriptions	394
9	General-Purpose Input/Outputs (GPIOs)	423
9.1	Signal Description	423
9.2	Functional Description	427
9.2.1	Data Control	429
9.2.2	Interrupt Control	430
9.2.3	Mode Control	431
9.2.4	Commit Control	431
9.2.5	Pad Control	432
9.2.6	Identification	432
9.3	Initialization and Configuration	432
9.4	Register Map	433
9.5	Register Descriptions	436
10	General-Purpose Timers	479
10.1	Block Diagram	480
10.2	Signal Description	480
10.3	Functional Description	483
10.3.1	GPTM Reset Conditions	483
10.3.2	Timer Modes	484
10.3.3	DMA Operation	489
10.3.4	Accessing Concatenated Register Values	489
10.4	Initialization and Configuration	490
10.4.1	32-Bit One-Shot/Periodic Timer Mode	490
10.4.2	32-Bit Real-Time Clock (RTC) Mode	491
10.4.3	16-Bit One-Shot/Periodic Timer Mode	491
10.4.4	Input Edge-Count Mode	492
10.4.5	16-Bit Input Edge Timing Mode	492
10.4.6	16-Bit PWM Mode	493
10.5	Register Map	493
10.6	Register Descriptions	494
11	Watchdog Timers	525
11.1	Block Diagram	526
11.2	Functional Description	526
11.2.1	Register Access Timing	527
11.3	Initialization and Configuration	527
11.4	Register Map	527
11.5	Register Descriptions	528
12	Analog-to-Digital Converter (ADC)	550
12.1	Block Diagram	551

12.2	Signal Description	552
12.3	Functional Description	554
12.3.1	Sample Sequencers	554
12.3.2	Module Control	555
12.3.3	Hardware Sample Averaging Circuit	557
12.3.4	Analog-to-Digital Converter	557
12.3.5	Differential Sampling	559
12.3.6	Internal Temperature Sensor	562
12.3.7	Digital Comparator Unit	563
12.4	Initialization and Configuration	567
12.4.1	Module Initialization	567
12.4.2	Sample Sequencer Configuration	568
12.5	Register Map	568
12.6	Register Descriptions	570
13	Universal Asynchronous Receivers/Transmitters (UARTs)	627
13.1	Block Diagram	628
13.2	Signal Description	628
13.3	Functional Description	630
13.3.1	Transmit/Receive Logic	630
13.3.2	Baud-Rate Generation	631
13.3.3	Data Transmission	632
13.3.4	Serial IR (SIR)	632
13.3.5	ISO 7816 Support	633
13.3.6	Modem Handshake Support	633
13.3.7	LIN Support	635
13.3.8	FIFO Operation	636
13.3.9	Interrupts	636
13.3.10	Loopback Operation	637
13.3.11	DMA Operation	637
13.4	Initialization and Configuration	638
13.5	Register Map	639
13.6	Register Descriptions	640
14	Synchronous Serial Interface (SSI)	688
14.1	Block Diagram	689
14.2	Signal Description	689
14.3	Functional Description	690
14.3.1	Bit Rate Generation	691
14.3.2	FIFO Operation	691
14.3.3	Interrupts	691
14.3.4	Frame Formats	692
14.3.5	DMA Operation	700
14.4	Initialization and Configuration	700
14.5	Register Map	702
14.6	Register Descriptions	703
15	Inter-Integrated Circuit (I²C) Interface	731
15.1	Block Diagram	732
15.2	Signal Description	732
15.3	Functional Description	733

15.3.1	I ² C Bus Functional Overview	733
15.3.2	Available Speed Modes	735
15.3.3	Interrupts	736
15.3.4	Loopback Operation	737
15.3.5	Command Sequence Flow Charts	737
15.4	Initialization and Configuration	744
15.5	Register Map	745
15.6	Register Descriptions (I ² C Master)	746
15.7	Register Descriptions (I ² C Slave)	758
16	Inter-Integrated Circuit Sound (I²S) Interface	767
16.1	Block Diagram	768
16.2	Signal Description	768
16.3	Functional Description	769
16.3.1	Transmit	771
16.3.2	Receive	775
16.4	Initialization and Configuration	777
16.5	Register Map	778
16.6	Register Descriptions	779
17	Controller Area Network (CAN) Module	803
17.1	Block Diagram	804
17.2	Signal Description	804
17.3	Functional Description	805
17.3.1	Initialization	806
17.3.2	Operation	807
17.3.3	Transmitting Message Objects	808
17.3.4	Configuring a Transmit Message Object	808
17.3.5	Updating a Transmit Message Object	809
17.3.6	Accepting Received Message Objects	810
17.3.7	Receiving a Data Frame	810
17.3.8	Receiving a Remote Frame	810
17.3.9	Receive/Transmit Priority	811
17.3.10	Configuring a Receive Message Object	811
17.3.11	Handling of Received Message Objects	812
17.3.12	Handling of Interrupts	814
17.3.13	Test Mode	815
17.3.14	Bit Timing Configuration Error Considerations	817
17.3.15	Bit Time and Bit Rate	817
17.3.16	Calculating the Bit Timing Parameters	819
17.4	Register Map	822
17.5	CAN Register Descriptions	823
18	Ethernet Controller	853
18.1	Block Diagram	854
18.2	Signal Description	855
18.3	Functional Description	856
18.3.1	MAC Operation	856
18.3.2	Internal MII Operation	859
18.3.3	PHY Operation	860

18.3.4	Interrupts	862
18.3.5	DMA Operation	863
18.4	Initialization and Configuration	863
18.4.1	Hardware Configuration	863
18.4.2	Software Configuration	864
18.5	Register Map	865
18.6	Ethernet MAC Register Descriptions	867
18.7	MII Management Register Descriptions	893
19	Universal Serial Bus (USB) Controller	914
19.1	Block Diagram	915
19.2	Signal Description	915
19.3	Functional Description	917
19.3.1	Operation as a Device	917
19.3.2	Operation as a Host	922
19.3.3	OTG Mode	926
19.3.4	DMA Operation	928
19.4	Initialization and Configuration	929
19.4.1	Pin Configuration	929
19.4.2	Endpoint Configuration	929
19.5	Register Map	930
19.6	Register Descriptions	941
20	Analog Comparators	1053
20.1	Block Diagram	1053
20.2	Signal Description	1054
20.3	Functional Description	1055
20.3.1	Internal Reference Programming	1055
20.4	Initialization and Configuration	1057
20.5	Register Map	1057
20.6	Register Descriptions	1057
21	Pulse Width Modulator (PWM)	1065
21.1	Block Diagram	1066
21.2	Signal Description	1067
21.3	Functional Description	1070
21.3.1	PWM Timer	1070
21.3.2	PWM Comparators	1070
21.3.3	PWM Signal Generator	1071
21.3.4	Dead-Band Generator	1072
21.3.5	Interrupt/ADC-Trigger Selector	1073
21.3.6	Synchronization Methods	1073
21.3.7	Fault Conditions	1074
21.3.8	Output Control Block	1075
21.4	Initialization and Configuration	1075
21.5	Register Map	1076
21.6	Register Descriptions	1078
22	Quadrature Encoder Interface (QEI)	1137
22.1	Block Diagram	1137
22.2	Signal Description	1138

22.3	Functional Description	1139
22.4	Initialization and Configuration	1141
22.5	Register Map	1142
22.6	Register Descriptions	1143
23	Pin Diagram	1160
24	Signal Tables	1162
24.1	100-Pin LQFP Package Pin Tables	1163
24.2	108-Pin BGA Package Pin Tables	1195
24.3	Connections for Unused Signals	1228
25	Operating Characteristics	1231
26	Electrical Characteristics	1232
26.1	DC Characteristics	1232
26.1.1	Maximum Ratings	1232
26.1.2	Recommended DC Operating Conditions	1232
26.1.3	On-Chip Low Drop-Out (LDO) Regulator Characteristics	1233
26.1.4	Hibernation Module Characteristics	1233
26.1.5	Flash Memory Characteristics	1233
26.1.6	GPIO Module Characteristics	1234
26.1.7	USB Module Characteristics	1234
26.1.8	Ethernet Controller Characteristics	1234
26.1.9	Current Specifications	1234
26.2	AC Characteristics	1235
26.2.1	Load Conditions	1235
26.2.2	Clocks	1235
26.2.3	JTAG and Boundary Scan	1238
26.2.4	Reset	1239
26.2.5	Sleep Modes	1241
26.2.6	Hibernation Module	1241
26.2.7	General-Purpose I/O (GPIO)	1243
26.2.8	Analog-to-Digital Converter (ADC)	1243
26.2.9	Synchronous Serial Interface (SSI)	1245
26.2.10	Inter-Integrated Circuit (I^2C) Interface	1246
26.2.11	Inter-Integrated Circuit Sound (I^2S) Interface	1247
26.2.12	Ethernet Controller	1249
26.2.13	Universal Serial Bus (USB) Controller	1251
26.2.14	Analog Comparator	1252
A	Register Quick Reference	1253
B	Ordering and Contact Information	1304
B.1	Ordering Information	1304
B.2	Part Markings	1304
B.3	Kits	1305
B.4	Support Information	1305
C	Package Information	1306
C.1	100-Pin LQFP Package	1306
C.1.1	Package Dimensions	1306
C.1.2	Tray Dimensions	1308

C.1.3	Tape and Reel Dimensions	1308
C.2	108-Ball BGA Package	1310
C.2.1	Package Dimensions	1310
C.2.2	Tray Dimensions	1312
C.2.3	Tape and Reel Dimensions	1313

List of Figures

Figure 1-1.	Stellaris LM3S9L97 Microcontroller High-Level Block Diagram	71
Figure 2-1.	CPU Block Diagram	75
Figure 2-2.	TPIU Block Diagram	76
Figure 2-3.	Cortex-M3 Register Set	78
Figure 2-4.	Bit-Band Mapping	99
Figure 2-5.	Data Storage	100
Figure 2-6.	Vector table	106
Figure 2-7.	Exception Stack Frame	108
Figure 3-1.	SRD Use Example	122
Figure 4-1.	JTAG Module Block Diagram	183
Figure 4-2.	Test Access Port State Machine	186
Figure 4-3.	IDCODE Register Format	192
Figure 4-4.	BYPASS Register Format	192
Figure 4-5.	Boundary Scan Register Format	193
Figure 5-1.	Basic \bar{RST} Configuration	197
Figure 5-2.	External Circuitry to Extend Power-On Reset	197
Figure 5-3.	Reset Circuit Controlled by Switch	198
Figure 5-4.	Power Architecture	203
Figure 5-5.	Main Clock Tree	206
Figure 6-1.	Hibernation Module Block Diagram	302
Figure 6-2.	Using a Crystal as the Hibernation Clock Source	305
Figure 6-3.	Using a Dedicated Oscillator as the Hibernation Clock Source with VDD3ON Mode	305
Figure 7-1.	Internal Memory Block Diagram	328
Figure 8-1.	μ DMA Block Diagram	366
Figure 8-2.	Example of Ping-Pong μ DMA Transaction	372
Figure 8-3.	Memory Scatter-Gather, Setup and Configuration	374
Figure 8-4.	Memory Scatter-Gather, μ DMA Copy Sequence	375
Figure 8-5.	Peripheral Scatter-Gather, Setup and Configuration	377
Figure 8-6.	Peripheral Scatter-Gather, μ DMA Copy Sequence	378
Figure 9-1.	Digital I/O Pads	428
Figure 9-2.	Analog/Digital I/O Pads	429
Figure 9-3.	GPIODATA Write Example	430
Figure 9-4.	GPIODATA Read Example	430
Figure 10-1.	GPTM Module Block Diagram	480
Figure 10-2.	Timer Daisy Chain	485
Figure 10-3.	Input Edge-Count Mode Example	487
Figure 10-4.	16-Bit Input Edge-Time Mode Example	488
Figure 10-5.	16-Bit PWM Mode Example	489
Figure 11-1.	WDT Module Block Diagram	526
Figure 12-1.	Implementation of Two ADC Blocks	551
Figure 12-2.	ADC Module Block Diagram	552
Figure 12-3.	ADC Sample Phases	556
Figure 12-4.	Doubling the ADC Sample Rate	556
Figure 12-5.	Skewed Sampling	557
Figure 12-6.	Internal Voltage Conversion Result	558

Figure 12-7. External Voltage Conversion Result	559
Figure 12-8. Differential Sampling Range, $V_{IN_ODD} = 1.5\text{ V}$	561
Figure 12-9. Differential Sampling Range, $V_{IN_ODD} = 0.75\text{ V}$	561
Figure 12-10. Differential Sampling Range, $V_{IN_ODD} = 2.25\text{ V}$	562
Figure 12-11. Internal Temperature Sensor Characteristic	563
Figure 12-12. Low-Band Operation (CIC=0x0 and/or CTC=0x0)	565
Figure 12-13. Mid-Band Operation (CIC=0x1 and/or CTC=0x1)	566
Figure 12-14. High-Band Operation (CIC=0x3 and/or CTC=0x3)	567
Figure 13-1. UART Module Block Diagram	628
Figure 13-2. UART Character Frame	631
Figure 13-3. IrDA Data Modulation	633
Figure 13-4. LIN Message	635
Figure 13-5. LIN Synchronization Field	636
Figure 14-1. SSI Module Block Diagram	689
Figure 14-2. TI Synchronous Serial Frame Format (Single Transfer)	693
Figure 14-3. TI Synchronous Serial Frame Format (Continuous Transfer)	694
Figure 14-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0	694
Figure 14-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0	695
Figure 14-6. Freescale SPI Frame Format with SPO=0 and SPH=1	696
Figure 14-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0	696
Figure 14-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0	697
Figure 14-9. Freescale SPI Frame Format with SPO=1 and SPH=1	698
Figure 14-10. MICROWIRE Frame Format (Single Frame)	698
Figure 14-11. MICROWIRE Frame Format (Continuous Transfer)	699
Figure 14-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements	700
Figure 15-1. I ² C Block Diagram	732
Figure 15-2. I ² C Bus Configuration	733
Figure 15-3. START and STOP Conditions	734
Figure 15-4. Complete Data Transfer with a 7-Bit Address	734
Figure 15-5. R/S Bit in First Byte	734
Figure 15-6. Data Validity During Bit Transfer on the I ² C Bus	735
Figure 15-7. Master Single TRANSMIT	738
Figure 15-8. Master Single RECEIVE	739
Figure 15-9. Master TRANSMIT with Repeated START	740
Figure 15-10. Master RECEIVE with Repeated START	741
Figure 15-11. Master RECEIVE with Repeated START after TRANSMIT with Repeated START	742
Figure 15-12. Master TRANSMIT with Repeated START after RECEIVE with Repeated START	743
Figure 15-13. Slave Command Sequence	744
Figure 16-1. I ² S Block Diagram	768
Figure 16-2. I ² S Data Transfer	771
Figure 16-3. Left-Justified Data Transfer	771
Figure 16-4. Right-Justified Data Transfer	771
Figure 17-1. CAN Controller Block Diagram	804
Figure 17-2. CAN Data/Remote Frame	806
Figure 17-3. Message Objects in a FIFO Buffer	814
Figure 17-4. CAN Bit Time	818

Figure 18-1.	Ethernet Controller	854
Figure 18-2.	Ethernet Controller Block Diagram	854
Figure 18-3.	Ethernet Frame	856
Figure 18-4.	Interface to an Ethernet Jack	864
Figure 19-1.	USB Module Block Diagram	915
Figure 20-1.	Analog Comparator Module Block Diagram	1053
Figure 20-2.	Structure of Comparator Unit	1055
Figure 20-3.	Comparator Internal Reference Structure	1056
Figure 21-1.	PWM Unit Diagram	1067
Figure 21-2.	PWM Module Block Diagram	1067
Figure 21-3.	PWM Count-Down Mode	1071
Figure 21-4.	PWM Count-Up/Down Mode	1071
Figure 21-5.	PWM Generation Example In Count-Up/Down Mode	1072
Figure 21-6.	PWM Dead-Band Generator	1072
Figure 22-1.	QEI Block Diagram	1138
Figure 22-2.	Quadrature Encoder and Velocity Predivider Operation	1140
Figure 23-1.	100-Pin LQFP Package Pin Diagram	1160
Figure 23-2.	108-Ball BGA Package Pin Diagram (Top View)	1161
Figure 26-1.	Load Conditions	1235
Figure 26-2.	JTAG Test Clock Input Timing	1239
Figure 26-3.	JTAG Test Access Port (TAP) Timing	1239
Figure 26-4.	Power-On Reset and Voltage Parameters	1240
Figure 26-5.	External Reset Timing (<u>RST</u>)	1240
Figure 26-6.	Power-On Reset Timing	1240
Figure 26-7.	Brown-Out Reset Timing	1240
Figure 26-8.	Software Reset Timing	1241
Figure 26-9.	Watchdog Reset Timing	1241
Figure 26-10.	MOSC Failure Reset Timing	1241
Figure 26-11.	Hibernation Module Timing with Internal Oscillator Running in Hibernation	1242
Figure 26-12.	Hibernation Module Timing with Internal Oscillator Stopped in Hibernation	1242
Figure 26-13.	VDD Ramp when Waking from Hibernation	1243
Figure 26-14.	ADC Input Equivalency Diagram	1244
Figure 26-15.	SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement	1245
Figure 26-16.	SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer	1246
Figure 26-17.	SSI Timing for SPI Frame Format (FRF=00), with SPH=1	1246
Figure 26-18.	I ² C Timing	1247
Figure 26-19.	I ² S Master Mode Transmit Timing	1248
Figure 26-20.	I ² S Master Mode Receive Timing	1248
Figure 26-21.	I ² S Slave Mode Transmit Timing	1249
Figure 26-22.	I ² S Slave Mode Receive Timing	1249
Figure 26-23.	External XTLP Oscillator Characteristics	1251
Figure C-1.	100-Pin LQFP Package Dimensions	1306
Figure C-2.	100-Pin LQFP Tray Dimensions	1308
Figure C-3.	100-Pin LQFP Tape and Reel Dimensions	1309
Figure C-4.	108-Ball BGA Package Dimensions	1310
Figure C-5.	108-Ball BGA Tray Dimensions	1312
Figure C-6.	108-Ball BGA Tape and Reel Dimensions	1313

List of Tables

Table 1.	Revision History	39
Table 2.	Documentation Conventions	47
Table 2-1.	Summary of Processor Mode, Privilege Level, and Stack Use	78
Table 2-2.	Processor Register Map	79
Table 2-3.	PSR Register Combinations	84
Table 2-4.	Memory Map	92
Table 2-5.	Memory Access Behavior	95
Table 2-6.	SRAM Memory Bit-Banding Regions	97
Table 2-7.	Peripheral Memory Bit-Banding Regions	97
Table 2-8.	Exception Types	103
Table 2-9.	Interrupts	104
Table 2-10.	Exception Return Behavior	109
Table 2-11.	Faults	109
Table 2-12.	Fault Status and Fault Address Registers	111
Table 2-13.	Cortex-M3 Instruction Summary	113
Table 3-1.	Core Peripheral Register Regions	116
Table 3-2.	Memory Attributes Summary	119
Table 3-3.	TEX, S, C, and B Bit Field Encoding	122
Table 3-4.	Cache Policy for Memory Attribute Encoding	123
Table 3-5.	AP Bit Field Encoding	123
Table 3-6.	Memory Region Attributes for Stellaris Microcontrollers	123
Table 3-7.	Peripherals Register Map	124
Table 3-8.	Interrupt Priority Levels	151
Table 3-9.	Example SIZE Field Values	179
Table 4-1.	Signals for JTAG_SWD_SWO (100LQFP)	183
Table 4-2.	Signals for JTAG_SWD_SWO (108BGA)	184
Table 4-3.	JTAG Port Pins State after Power-On Reset or \overline{RST} assertion	185
Table 4-4.	JTAG Instruction Register Commands	190
Table 5-1.	Signals for System Control & Clocks (100LQFP)	194
Table 5-2.	Signals for System Control & Clocks (108BGA)	194
Table 5-3.	Reset Sources	195
Table 5-4.	Clock Source Options	204
Table 5-5.	Possible System Clock Frequencies Using the SYSDIV Field	207
Table 5-6.	Examples of Possible System Clock Frequencies Using the SYSDIV2 Field	207
Table 5-7.	Examples of Possible System Clock Frequencies with DIV400=1	208
Table 5-8.	System Control Register Map	213
Table 5-9.	RCC2 Fields that Override RCC Fields	234
Table 6-1.	Signals for Hibernate (100LQFP)	302
Table 6-2.	Signals for Hibernate (108BGA)	303
Table 6-3.	Hibernation Module Clock Operation	309
Table 6-4.	Hibernation Module Register Map	311
Table 7-1.	Flash Memory Protection Policy Combinations	332
Table 7-2.	User-Programmable Flash Memory Resident Registers	335
Table 7-3.	Flash Register Map	336
Table 8-1.	μ DMA Channel Assignments	367
Table 8-2.	Request Type Support	369

Table 8-3.	Control Structure Memory Map	370
Table 8-4.	Channel Control Structure	370
Table 8-5.	μDMA Read Example: 8-Bit Peripheral	379
Table 8-6.	μDMA Interrupt Assignments	380
Table 8-7.	Channel Control Structure Offsets for Channel 30	381
Table 8-8.	Channel Control Word Configuration for Memory Transfer Example	381
Table 8-9.	Channel Control Structure Offsets for Channel 7	382
Table 8-10.	Channel Control Word Configuration for Peripheral Transmit Example	383
Table 8-11.	Primary and Alternate Channel Control Structure Offsets for Channel 8	384
Table 8-12.	Channel Control Word Configuration for Peripheral Ping-Pong Receive Example	385
Table 8-13.	μDMA Register Map	386
Table 9-1.	GPIO Pins With Non-Zero Reset Values	424
Table 9-2.	GPIO Pins and Alternate Functions (100LQFP)	424
Table 9-3.	GPIO Pins and Alternate Functions (108BGA)	426
Table 9-4.	GPIO Pad Configuration Examples	432
Table 9-5.	GPIO Interrupt Configuration Example	433
Table 9-6.	GPIO Pins With Non-Zero Reset Values	434
Table 9-7.	GPIO Register Map	434
Table 9-8.	GPIO Pins With Non-Zero Reset Values	447
Table 9-9.	GPIO Pins With Non-Zero Reset Values	453
Table 9-10.	GPIO Pins With Non-Zero Reset Values	455
Table 9-11.	GPIO Pins With Non-Zero Reset Values	458
Table 9-12.	GPIO Pins With Non-Zero Reset Values	465
Table 10-1.	Available CCP Pins	480
Table 10-2.	Signals for General-Purpose Timers (100LQFP)	481
Table 10-3.	Signals for General-Purpose Timers (108BGA)	482
Table 10-4.	General-Purpose Timer Capabilities	483
Table 10-5.	16-Bit Timer With Prescaler Configurations	485
Table 10-6.	Timers Register Map	494
Table 11-1.	Watchdog Timers Register Map	528
Table 12-1.	Signals for ADC (100LQFP)	552
Table 12-2.	Signals for ADC (108BGA)	553
Table 12-3.	Samples and FIFO Depth of Sequencers	554
Table 12-4.	Differential Sampling Pairs	559
Table 12-5.	ADC Register Map	568
Table 13-1.	Signals for UART (100LQFP)	629
Table 13-2.	Signals for UART (108BGA)	629
Table 13-3.	Flow Control Mode	634
Table 13-4.	UART Register Map	639
Table 14-1.	Signals for SSI (100LQFP)	690
Table 14-2.	Signals for SSI (108BGA)	690
Table 14-3.	SSI Register Map	702
Table 15-1.	Signals for I ² C (100LQFP)	732
Table 15-2.	Signals for I ² C (108BGA)	732
Table 15-3.	Examples of I ² C Master Timer Period versus Speed Mode	736
Table 15-4.	Inter-Integrated Circuit (I ² C) Interface Register Map	745
Table 15-5.	Write Field Decoding for I2CMCS[3:0] Field	750

Table 16-1.	Signals for I ² S (100LQFP)	769
Table 16-2.	Signals for I ² S (108BGA)	769
Table 16-3.	I ² S Transmit FIFO Interface	772
Table 16-4.	Crystal Frequency (Values from 3.5795 MHz to 5 MHz)	773
Table 16-5.	Crystal Frequency (Values from 5.12 MHz to 8.192 MHz)	773
Table 16-6.	Crystal Frequency (Values from 10 MHz to 14.3181 MHz)	774
Table 16-7.	Crystal Frequency (Values from 16 MHz to 16.384 MHz)	774
Table 16-8.	I ² S Receive FIFO Interface	776
Table 16-9.	Audio Formats Configuration	778
Table 16-10.	Inter-Integrated Circuit Sound (I ² S) Interface Register Map	779
Table 17-1.	Signals for Controller Area Network (100LQFP)	805
Table 17-2.	Signals for Controller Area Network (108BGA)	805
Table 17-3.	Message Object Configurations	811
Table 17-4.	CAN Protocol Ranges	818
Table 17-5.	CANBIT Register Values	818
Table 17-6.	CAN Register Map	822
Table 18-1.	Signals for Ethernet (100LQFP)	855
Table 18-2.	Signals for Ethernet (108BGA)	855
Table 18-3.	TX & RX FIFO Organization	858
Table 18-4.	Ethernet Register Map	865
Table 19-1.	Signals for USB (100LQFP)	915
Table 19-2.	Signals for USB (108BGA)	916
Table 19-3.	Remainder (MAXLOAD/4)	928
Table 19-4.	Actual Bytes Read	928
Table 19-5.	Packet Sizes That Clear RXRDY	928
Table 19-6.	Universal Serial Bus (USB) Controller Register Map	930
Table 20-1.	Signals for Analog Comparators (100LQFP)	1054
Table 20-2.	Signals for Analog Comparators (108BGA)	1054
Table 20-3.	Internal Reference Voltage and AREFCTL Field Values	1056
Table 20-4.	Analog Comparators Register Map	1057
Table 21-1.	Signals for PWM (100LQFP)	1068
Table 21-2.	Signals for PWM (108BGA)	1069
Table 21-3.	PWM Register Map	1076
Table 22-1.	Signals for QEI (100LQFP)	1138
Table 22-2.	Signals for QEI (108BGA)	1139
Table 22-3.	QEI Register Map	1142
Table 24-1.	GPIO Pins With Default Alternate Functions	1162
Table 24-2.	Signals by Pin Number	1163
Table 24-3.	Signals by Signal Name	1173
Table 24-4.	Signals by Function, Except for GPIO	1182
Table 24-5.	GPIO Pins and Alternate Functions	1190
Table 24-6.	Possible Pin Assignments for Alternate Functions	1193
Table 24-7.	Signals by Pin Number	1195
Table 24-8.	Signals by Signal Name	1206
Table 24-9.	Signals by Function, Except for GPIO	1215
Table 24-10.	GPIO Pins and Alternate Functions	1223
Table 24-11.	Possible Pin Assignments for Alternate Functions	1226
Table 24-12.	Connections for Unused Signals (100-pin LQFP)	1228

Table 24-13.	Connections for Unused Signals, 108-pin BGA	1229
Table 25-1.	Temperature Characteristics	1231
Table 25-2.	Thermal Characteristics	1231
Table 25-3.	ESD Absolute Maximum Ratings	1231
Table 26-1.	Maximum Ratings	1232
Table 26-2.	Recommended DC Operating Conditions	1232
Table 26-3.	LDO Regulator Characteristics	1233
Table 26-4.	Hibernation Module DC Characteristics	1233
Table 26-5.	Flash Memory Characteristics	1233
Table 26-6.	GPIO Module DC Characteristics	1234
Table 26-7.	USB Controller DC Characteristics	1234
Table 26-8.	Ethernet Controller DC Characteristics	1234
Table 26-9.	Preliminary Current Consumption	1234
Table 26-10.	Phase Locked Loop (PLL) Characteristics	1236
Table 26-11.	Actual PLL Frequency	1236
Table 26-12.	PIOSC Clock Characteristics	1236
Table 26-13.	30-kHz Clock Characteristics	1237
Table 26-14.	Hibernation Clock Characteristics	1237
Table 26-15.	HIB Oscillator Input Characteristics	1237
Table 26-16.	Main Oscillator Clock Characteristics	1237
Table 26-17.	MOSC Oscillator Input Characteristics	1238
Table 26-18.	System Clock Characteristics with ADC Operation	1238
Table 26-19.	JTAG Characteristics	1238
Table 26-20.	Reset Characteristics	1239
Table 26-21.	Sleep Modes AC Characteristics	1241
Table 26-22.	Hibernation Module AC Characteristics	1242
Table 26-23.	GPIO Characteristics	1243
Table 26-24.	ADC Characteristics	1243
Table 26-25.	ADC Module External Reference Characteristics	1244
Table 26-26.	ADC Module Internal Reference Characteristics	1245
Table 26-27.	SSI Characteristics	1245
Table 26-28.	I ² C Characteristics	1246
Table 26-29.	I ² S Master Clock (Receive and Transmit)	1247
Table 26-30.	I ² S Slave Clock (Receive and Transmit)	1247
Table 26-31.	I ² S Master Mode	1248
Table 26-32.	I ² S Slave Mode	1248
Table 26-33.	100BASE-TX Transmitter Characteristics	1249
Table 26-34.	100BASE-TX Transmitter Characteristics (informative)	1249
Table 26-35.	100BASE-TX Receiver Characteristics	1249
Table 26-36.	10BASE-T Transmitter Characteristics	1250
Table 26-37.	10BASE-T Transmitter Characteristics (informative)	1250
Table 26-38.	10BASE-T Receiver Characteristics	1250
Table 26-39.	Isolation Transformers	1250
Table 26-40.	Ethernet Reference Crystal	1251
Table 26-41.	External XTLP Oscillator Characteristics	1251
Table 26-42.	Analog Comparator Characteristics	1252
Table 26-43.	Analog Comparator Voltage Reference Characteristics	1252
Table B-1.	Part Ordering Information	1304

List of Registers

The Cortex-M3 Processor	73
Register 1: Cortex General-Purpose Register 0 (R0)	80
Register 2: Cortex General-Purpose Register 1 (R1)	80
Register 3: Cortex General-Purpose Register 2 (R2)	80
Register 4: Cortex General-Purpose Register 3 (R3)	80
Register 5: Cortex General-Purpose Register 4 (R4)	80
Register 6: Cortex General-Purpose Register 5 (R5)	80
Register 7: Cortex General-Purpose Register 6 (R6)	80
Register 8: Cortex General-Purpose Register 7 (R7)	80
Register 9: Cortex General-Purpose Register 8 (R8)	80
Register 10: Cortex General-Purpose Register 9 (R9)	80
Register 11: Cortex General-Purpose Register 10 (R10)	80
Register 12: Cortex General-Purpose Register 11 (R11)	80
Register 13: Cortex General-Purpose Register 12 (R12)	80
Register 14: Stack Pointer (SP)	81
Register 15: Link Register (LR)	82
Register 16: Program Counter (PC)	83
Register 17: Program Status Register (PSR)	84
Register 18: Priority Mask Register (PRIMASK)	88
Register 19: Fault Mask Register (FAULTMASK)	89
Register 20: Base Priority Mask Register (BASEPRI)	90
Register 21: Control Register (CONTROL)	91
Cortex-M3 Peripherals	116
Register 1: SysTick Control and Status Register (STCTRL), offset 0x010	127
Register 2: SysTick Reload Value Register (STRELOAD), offset 0x014	129
Register 3: SysTick Current Value Register (STCURRENT), offset 0x018	130
Register 4: Interrupt 0-31 Set Enable (EN0), offset 0x100	131
Register 5: Interrupt 32-54 Set Enable (EN1), offset 0x104	132
Register 6: Interrupt 0-31 Clear Enable (DIS0), offset 0x180	133
Register 7: Interrupt 32-54 Clear Enable (DIS1), offset 0x184	134
Register 8: Interrupt 0-31 Set Pending (PEND0), offset 0x200	135
Register 9: Interrupt 32-54 Set Pending (PEND1), offset 0x204	136
Register 10: Interrupt 0-31 Clear Pending (UNPEND0), offset 0x280	137
Register 11: Interrupt 32-54 Clear Pending (UNPEND1), offset 0x284	138
Register 12: Interrupt 0-31 Active Bit (ACTIVE0), offset 0x300	139
Register 13: Interrupt 32-54 Active Bit (ACTIVE1), offset 0x304	140
Register 14: Interrupt 0-3 Priority (PRI0), offset 0x400	141
Register 15: Interrupt 4-7 Priority (PRI1), offset 0x404	141
Register 16: Interrupt 8-11 Priority (PRI2), offset 0x408	141
Register 17: Interrupt 12-15 Priority (PRI3), offset 0x40C	141
Register 18: Interrupt 16-19 Priority (PRI4), offset 0x410	141
Register 19: Interrupt 20-23 Priority (PRI5), offset 0x414	141
Register 20: Interrupt 24-27 Priority (PRI6), offset 0x418	141
Register 21: Interrupt 28-31 Priority (PRI7), offset 0x41C	141
Register 22: Interrupt 32-35 Priority (PRI8), offset 0x420	141

Register 23:	Interrupt 36-39 Priority (PRI9), offset 0x424	141
Register 24:	Interrupt 40-43 Priority (PRI10), offset 0x428	141
Register 25:	Interrupt 44-47 Priority (PRI11), offset 0x42C	141
Register 26:	Interrupt 48-51 Priority (PRI12), offset 0x430	141
Register 27:	Interrupt 52-54 Priority (PRI13), offset 0x434	141
Register 28:	Software Trigger Interrupt (SWTRIG), offset 0xF00	143
Register 29:	Auxiliary Control (ACTLR), offset 0x008	144
Register 30:	CPU ID Base (CPUID), offset 0xD00	146
Register 31:	Interrupt Control and State (INTCTRL), offset 0xD04	147
Register 32:	Vector Table Offset (VTABLE), offset 0xD08	150
Register 33:	Application Interrupt and Reset Control (APINT), offset 0xD0C	151
Register 34:	System Control (SYSCTRL), offset 0xD10	153
Register 35:	Configuration and Control (CFGCTRL), offset 0xD14	155
Register 36:	System Handler Priority 1 (SYSPRI1), offset 0xD18	157
Register 37:	System Handler Priority 2 (SYSPRI2), offset 0xD1C	158
Register 38:	System Handler Priority 3 (SYSPRI3), offset 0xD20	159
Register 39:	System Handler Control and State (SYSHNDCTRL), offset 0xD24	160
Register 40:	Configurable Fault Status (FAULTSTAT), offset 0xD28	164
Register 41:	Hard Fault Status (HFAULTSTAT), offset 0xD2C	170
Register 42:	Memory Management Fault Address (MMADDR), offset 0xD34	171
Register 43:	Bus Fault Address (FAULTADDR), offset 0xD38	172
Register 44:	MPU Type (MPUTYPE), offset 0xD90	173
Register 45:	MPU Control (MPUCTRL), offset 0xD94	174
Register 46:	MPU Region Number (MPUNUMBER), offset 0xD98	176
Register 47:	MPU Region Base Address (MPUBASE), offset 0xD9C	177
Register 48:	MPU Region Base Address Alias 1 (MPUBASE1), offset 0xDA4	177
Register 49:	MPU Region Base Address Alias 2 (MPUBASE2), offset 0xDAC	177
Register 50:	MPU Region Base Address Alias 3 (MPUBASE3), offset 0xDB4	177
Register 51:	MPU Region Attribute and Size (MPUATTR), offset 0xDA0	179
Register 52:	MPU Region Attribute and Size Alias 1 (MPUATTR1), offset 0xDA8	179
Register 53:	MPU Region Attribute and Size Alias 2 (MPUATTR2), offset 0xDB0	179
Register 54:	MPU Region Attribute and Size Alias 3 (MPUATTR3), offset 0xDB8	179
System Control	194	
Register 1:	Device Identification 0 (DID0), offset 0x000	215
Register 2:	Brown-Out Reset Control (PBORCTL), offset 0x030	217
Register 3:	Raw Interrupt Status (RIS), offset 0x050	218
Register 4:	Interrupt Mask Control (IMC), offset 0x054	220
Register 5:	Masked Interrupt Status and Clear (MISC), offset 0x058	222
Register 6:	Reset Cause (RESC), offset 0x05C	224
Register 7:	Run-Mode Clock Configuration (RCC), offset 0x060	226
Register 8:	XTAL to PLL Translation (PLLCFG), offset 0x064	231
Register 9:	GPIO High-Performance Bus Control (GPIOHBCtl), offset 0x06C	232
Register 10:	Run-Mode Clock Configuration 2 (RCC2), offset 0x070	234
Register 11:	Main Oscillator Control (MOSCCTL), offset 0x07C	237
Register 12:	Deep Sleep Clock Configuration (DSLPCLKCFG), offset 0x144	238
Register 13:	Precision Internal Oscillator Calibration (PIOSCCAL), offset 0x150	240
Register 14:	Precision Internal Oscillator Statistics (PIOSCSTAT), offset 0x154	242
Register 15:	I ² S MCLK Configuration (I2SMCLKCFG), offset 0x170	243

Register 16:	Device Identification 1 (DID1), offset 0x004	245
Register 17:	Device Capabilities 0 (DC0), offset 0x008	247
Register 18:	Device Capabilities 1 (DC1), offset 0x010	248
Register 19:	Device Capabilities 2 (DC2), offset 0x014	250
Register 20:	Device Capabilities 3 (DC3), offset 0x018	252
Register 21:	Device Capabilities 4 (DC4), offset 0x01C	254
Register 22:	Device Capabilities 5 (DC5), offset 0x020	256
Register 23:	Device Capabilities 6 (DC6), offset 0x024	258
Register 24:	Device Capabilities 7 (DC7), offset 0x028	259
Register 25:	Device Capabilities 8 ADC Channels (DC8), offset 0x02C	263
Register 26:	Device Capabilities 9 ADC Digital Comparators (DC9), offset 0x190	265
Register 27:	Non-Volatile Memory Information (NVSTAT), offset 0x1A0	267
Register 28:	Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100	268
Register 29:	Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110	271
Register 30:	Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120	274
Register 31:	Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104	276
Register 32:	Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114	279
Register 33:	Deep-Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124	282
Register 34:	Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108	285
Register 35:	Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118	288
Register 36:	Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128	291
Register 37:	Software Reset Control 0 (SRCR0), offset 0x040	294
Register 38:	Software Reset Control 1 (SRCR1), offset 0x044	296
Register 39:	Software Reset Control 2 (SRCR2), offset 0x048	299
Hibernation Module	301	
Register 1:	Hibernation RTC Counter (HIBRTCC), offset 0x000	312
Register 2:	Hibernation RTC Match 0 (HIBRTCM0), offset 0x004	313
Register 3:	Hibernation RTC Match 1 (HIBRTCM1), offset 0x008	314
Register 4:	Hibernation RTC Load (HIBRTCLD), offset 0x00C	315
Register 5:	Hibernation Control (HIBCTL), offset 0x010	316
Register 6:	Hibernation Interrupt Mask (HIBIM), offset 0x014	319
Register 7:	Hibernation Raw Interrupt Status (HIBRIS), offset 0x018	321
Register 8:	Hibernation Masked Interrupt Status (HIBMIS), offset 0x01C	323
Register 9:	Hibernation Interrupt Clear (HIBIC), offset 0x020	325
Register 10:	Hibernation RTC Trim (HIBRTCT), offset 0x024	326
Register 11:	Hibernation Data (HIBDATA), offset 0x030-0x12C	327
Internal Memory	328	
Register 1:	Flash Memory Address (FMA), offset 0x000	338
Register 2:	Flash Memory Data (FMD), offset 0x004	339
Register 3:	Flash Memory Control (FMC), offset 0x008	340
Register 4:	Flash Controller Raw Interrupt Status (FCRIS), offset 0x00C	343
Register 5:	Flash Controller Interrupt Mask (FCIM), offset 0x010	344
Register 6:	Flash Controller Masked Interrupt Status and Clear (FCMISC), offset 0x014	345
Register 7:	Flash Memory Control 2 (FMC2), offset 0x020	346
Register 8:	Flash Write Buffer Valid (FWBVAL), offset 0x030	347
Register 9:	Flash Control (FCTL), offset 0x0F8	348
Register 10:	Flash Write Buffer n (FWBn), offset 0x100 - 0x17C	349
Register 11:	ROM Control (RMCTL), offset 0x0F0	350

Register 12:	Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200	351
Register 13:	Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400	352
Register 14:	Boot Configuration (BOOTCFG), offset 0x1D0	353
Register 15:	User Register 0 (USER_REG0), offset 0x1E0	355
Register 16:	User Register 1 (USER_REG1), offset 0x1E4	356
Register 17:	User Register 2 (USER_REG2), offset 0x1E8	357
Register 18:	User Register 3 (USER_REG3), offset 0x1EC	358
Register 19:	Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204	359
Register 20:	Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208	360
Register 21:	Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C	361
Register 22:	Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404	362
Register 23:	Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408	363
Register 24:	Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C	364
Micro Direct Memory Access (μDMA)	365	
Register 1:	DMA Channel Source Address End Pointer (DMASRCENDP), offset 0x000	388
Register 2:	DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004	389
Register 3:	DMA Channel Control Word (DMACHCTL), offset 0x008	390
Register 4:	DMA Status (DMASTAT), offset 0x000	395
Register 5:	DMA Configuration (DMACFG), offset 0x004	397
Register 6:	DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008	398
Register 7:	DMA Alternate Channel Control Base Pointer (DMAALTBASE), offset 0x00C	399
Register 8:	DMA Channel Wait-on-Request Status (DMAWAITSTAT), offset 0x010	400
Register 9:	DMA Channel Software Request (DMASWREQ), offset 0x014	401
Register 10:	DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018	402
Register 11:	DMA Channel Useburst Clear (DMAUSEBURSTCLR), offset 0x01C	403
Register 12:	DMA Channel Request Mask Set (DMAREQMASKSET), offset 0x020	404
Register 13:	DMA Channel Request Mask Clear (DMAREQMASKCLR), offset 0x024	405
Register 14:	DMA Channel Enable Set (DMAENASET), offset 0x028	406
Register 15:	DMA Channel Enable Clear (DMAENACLR), offset 0x02C	407
Register 16:	DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030	408
Register 17:	DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034	409
Register 18:	DMA Channel Priority Set (DMAPRIOSSET), offset 0x038	410
Register 19:	DMA Channel Priority Clear (DMAPIOCLR), offset 0x03C	411
Register 20:	DMA Bus Error Clear (DMAERRCLR), offset 0x04C	412
Register 21:	DMA Channel Assignment (DMACHASGN), offset 0x500	413
Register 22:	DMA Peripheral Identification 0 (DMAPeriphID0), offset 0xFE0	414
Register 23:	DMA Peripheral Identification 1 (DMAPeriphID1), offset 0xFE4	415
Register 24:	DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8	416
Register 25:	DMA Peripheral Identification 3 (DMAPeriphID3), offset 0xFEC	417
Register 26:	DMA Peripheral Identification 4 (DMAPeriphID4), offset 0xFD0	418
Register 27:	DMA PrimeCell Identification 0 (DMACellID0), offset 0xFF0	419
Register 28:	DMA PrimeCell Identification 1 (DMACellID1), offset 0xFF4	420
Register 29:	DMA PrimeCell Identification 2 (DMACellID2), offset 0xFF8	421
Register 30:	DMA PrimeCell Identification 3 (DMACellID3), offset 0xFFC	422
General-Purpose Input/Outputs (GPIOs)	423	
Register 1:	GPIO Data (GPIODATA), offset 0x000	437
Register 2:	GPIO Direction (GPIODIR), offset 0x400	438
Register 3:	GPIO Interrupt Sense (GPIOIS), offset 0x404	439

Register 4:	GPIO Interrupt Both Edges (GPIOIBE), offset 0x408	440
Register 5:	GPIO Interrupt Event (GPIOIEV), offset 0x40C	441
Register 6:	GPIO Interrupt Mask (GPIOIM), offset 0x410	442
Register 7:	GPIO Raw Interrupt Status (GPIORIS), offset 0x414	443
Register 8:	GPIO Masked Interrupt Status (GPIOMIS), offset 0x418	444
Register 9:	GPIO Interrupt Clear (GPIOICR), offset 0x41C	446
Register 10:	GPIO Alternate Function Select (GPIOAFSEL), offset 0x420	447
Register 11:	GPIO 2-mA Drive Select (GPIODR2R), offset 0x500	449
Register 12:	GPIO 4-mA Drive Select (GPIODR4R), offset 0x504	450
Register 13:	GPIO 8-mA Drive Select (GPIODR8R), offset 0x508	451
Register 14:	GPIO Open Drain Select (GPIOODR), offset 0x50C	452
Register 15:	GPIO Pull-Up Select (GPIOPUR), offset 0x510	453
Register 16:	GPIO Pull-Down Select (GPIOPDR), offset 0x514	455
Register 17:	GPIO Slew Rate Control Select (GPIOSLR), offset 0x518	457
Register 18:	GPIO Digital Enable (GPIODEN), offset 0x51C	458
Register 19:	GPIO Lock (GPIOLOCK), offset 0x520	460
Register 20:	GPIO Commit (GPIOCR), offset 0x524	461
Register 21:	GPIO Analog Mode Select (GPIOAMSEL), offset 0x528	463
Register 22:	GPIO Port Control (GPIOPCTL), offset 0x52C	465
Register 23:	GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0	467
Register 24:	GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4	468
Register 25:	GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8	469
Register 26:	GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC	470
Register 27:	GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0	471
Register 28:	GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4	472
Register 29:	GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8	473
Register 30:	GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC	474
Register 31:	GPIO PrimeCell Identification 0 (GPIOPCellID0), offset 0xFF0	475
Register 32:	GPIO PrimeCell Identification 1 (GPIOPCellID1), offset 0xFF4	476
Register 33:	GPIO PrimeCell Identification 2 (GPIOPCellID2), offset 0xFF8	477
Register 34:	GPIO PrimeCell Identification 3 (GPIOPCellID3), offset 0xFFC	478
General-Purpose Timers	479	
Register 1:	GPTM Configuration (GPTMCFG), offset 0x000	495
Register 2:	GPTM Timer A Mode (GPTMTAMR), offset 0x004	496
Register 3:	GPTM Timer B Mode (GPTMTBMR), offset 0x008	498
Register 4:	GPTM Control (GPTMCTL), offset 0x00C	500
Register 5:	GPTM Interrupt Mask (GPTMIMR), offset 0x018	503
Register 6:	GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C	505
Register 7:	GPTM Masked Interrupt Status (GPTMMIS), offset 0x020	508
Register 8:	GPTM Interrupt Clear (GPTMICR), offset 0x024	511
Register 9:	GPTM Timer A Interval Load (GPTMTAILR), offset 0x028	513
Register 10:	GPTM Timer B Interval Load (GPTMTBILR), offset 0x02C	514
Register 11:	GPTM Timer A Match (GPTMTAMATCHR), offset 0x030	515
Register 12:	GPTM Timer B Match (GPTMTBMATCHR), offset 0x034	516
Register 13:	GPTM Timer A Prescale (GPTMTAPR), offset 0x038	517
Register 14:	GPTM Timer B Prescale (GPTMTBPR), offset 0x03C	518
Register 15:	GPTM TimerA Prescale Match (GPTMTAPMR), offset 0x040	519
Register 16:	GPTM TimerB Prescale Match (GPTMTBPMR), offset 0x044	520

Register 17:	GPTM Timer A (GPTMTAR), offset 0x048	521
Register 18:	GPTM Timer B (GPTMTBR), offset 0x04C	522
Register 19:	GPTM Timer A Value (GPTMTAV), offset 0x050	523
Register 20:	GPTM Timer B Value (GPTMTBV), offset 0x054	524
Watchdog Timers		525
Register 1:	Watchdog Load (WDTLOAD), offset 0x000	529
Register 2:	Watchdog Value (WDTVALUE), offset 0x004	530
Register 3:	Watchdog Control (WDTCTL), offset 0x008	531
Register 4:	Watchdog Interrupt Clear (WDTICR), offset 0x00C	533
Register 5:	Watchdog Raw Interrupt Status (WDTRIS), offset 0x010	534
Register 6:	Watchdog Masked Interrupt Status (WDTMIS), offset 0x014	535
Register 7:	Watchdog Test (WDTTEST), offset 0x418	536
Register 8:	Watchdog Lock (WDTLOCK), offset 0xC00	537
Register 9:	Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0	538
Register 10:	Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4	539
Register 11:	Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8	540
Register 12:	Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC	541
Register 13:	Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0	542
Register 14:	Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4	543
Register 15:	Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8	544
Register 16:	Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC	545
Register 17:	Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0	546
Register 18:	Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4	547
Register 19:	Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8	548
Register 20:	Watchdog PrimeCell Identification 3 (WDTPCellID3), offset 0xFFC	549
Analog-to-Digital Converter (ADC)		550
Register 1:	ADC Active Sample Sequencer (ADCACTSS), offset 0x000	571
Register 2:	ADC Raw Interrupt Status (ADCRIS), offset 0x004	572
Register 3:	ADC Interrupt Mask (ADCIM), offset 0x008	574
Register 4:	ADC Interrupt Status and Clear (ADCISC), offset 0x00C	576
Register 5:	ADC Overflow Status (ADCOSTAT), offset 0x010	579
Register 6:	ADC Event Multiplexer Select (ADCEMUX), offset 0x014	581
Register 7:	ADC Underflow Status (ADCUSTAT), offset 0x018	586
Register 8:	ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020	587
Register 9:	ADC Sample Phase Control (ADCSPC), offset 0x024	589
Register 10:	ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028	590
Register 11:	ADC Sample Averaging Control (ADCSAC), offset 0x030	592
Register 12:	ADC Digital Comparator Interrupt Status and Clear (ADCDCISC), offset 0x034	593
Register 13:	ADC Control (ADCCTL), offset 0x038	595
Register 14:	ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040	596
Register 15:	ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044	598
Register 16:	ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048	601
Register 17:	ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068	601
Register 18:	ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088	601
Register 19:	ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8	601
Register 20:	ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C	602
Register 21:	ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C	602
Register 22:	ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C	602

Register 23:	ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC	602
Register 24:	ADC Sample Sequence 0 Operation (ADCSSOP0), offset 0x050	604
Register 25:	ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0), offset 0x054	606
Register 26:	ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060	608
Register 27:	ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080	608
Register 28:	ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064	609
Register 29:	ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084	609
Register 30:	ADC Sample Sequence 1 Operation (ADCSSOP1), offset 0x070	611
Register 31:	ADC Sample Sequence 2 Operation (ADCSSOP2), offset 0x090	611
Register 32:	ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1), offset 0x074	612
Register 33:	ADC Sample Sequence 2 Digital Comparator Select (ADCSSDC2), offset 0x094	612
Register 34:	ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0	614
Register 35:	ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4	615
Register 36:	ADC Sample Sequence 3 Operation (ADCSSOP3), offset 0x0B0	616
Register 37:	ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3), offset 0x0B4	617
Register 38:	ADC Digital Comparator Reset Initial Conditions (ADCDCRIC), offset 0xD00	618
Register 39:	ADC Digital Comparator Control 0 (ADCDCCTL0), offset 0xE00	623
Register 40:	ADC Digital Comparator Control 1 (ADCDCCTL1), offset 0xE04	623
Register 41:	ADC Digital Comparator Control 2 (ADCDCCTL2), offset 0xE08	623
Register 42:	ADC Digital Comparator Control 3 (ADCDCCTL3), offset 0xE0C	623
Register 43:	ADC Digital Comparator Control 4 (ADCDCCTL4), offset 0xE10	623
Register 44:	ADC Digital Comparator Control 5 (ADCDCCTL5), offset 0xE14	623
Register 45:	ADC Digital Comparator Control 6 (ADCDCCTL6), offset 0xE18	623
Register 46:	ADC Digital Comparator Control 7 (ADCDCCTL7), offset 0xE1C	623
Register 47:	ADC Digital Comparator Range 0 (ADCDCCMP0), offset 0xE40	626
Register 48:	ADC Digital Comparator Range 1 (ADCDCCMP1), offset 0xE44	626
Register 49:	ADC Digital Comparator Range 2 (ADCDCCMP2), offset 0xE48	626
Register 50:	ADC Digital Comparator Range 3 (ADCDCCMP3), offset 0xE4C	626
Register 51:	ADC Digital Comparator Range 4 (ADCDCCMP4), offset 0xE50	626
Register 52:	ADC Digital Comparator Range 5 (ADCDCCMP5), offset 0xE54	626
Register 53:	ADC Digital Comparator Range 6 (ADCDCCMP6), offset 0xE58	626
Register 54:	ADC Digital Comparator Range 7 (ADCDCCMP7), offset 0xE5C	626
Universal Asynchronous Receivers/Transmitters (UARTs)		627
Register 1:	UART Data (UARTDR), offset 0x000	641
Register 2:	UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004	643
Register 3:	UART Flag (UARTFR), offset 0x018	646
Register 4:	UART IrDA Low-Power Register (UARTILPR), offset 0x020	649
Register 5:	UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024	650
Register 6:	UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028	651
Register 7:	UART Line Control (UARTLCRH), offset 0x02C	652
Register 8:	UART Control (UARTCTL), offset 0x030	654
Register 9:	UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034	658
Register 10:	UART Interrupt Mask (UARTIM), offset 0x038	660
Register 11:	UART Raw Interrupt Status (UARTRIS), offset 0x03C	664
Register 12:	UART Masked Interrupt Status (UARTMIS), offset 0x040	667
Register 13:	UART Interrupt Clear (UARTICR), offset 0x044	670
Register 14:	UART DMA Control (UARTDMACTL), offset 0x048	672
Register 15:	UART LIN Control (UARTLCTL), offset 0x090	673

Register 16:	UART LIN Snap Shot (UARTLSS), offset 0x094	674
Register 17:	UART LIN Timer (UARTLTIM), offset 0x098	675
Register 18:	UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0	676
Register 19:	UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4	677
Register 20:	UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8	678
Register 21:	UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC	679
Register 22:	UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0	680
Register 23:	UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4	681
Register 24:	UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8	682
Register 25:	UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC	683
Register 26:	UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0	684
Register 27:	UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4	685
Register 28:	UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8	686
Register 29:	UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC	687
Synchronous Serial Interface (SSI)		688
Register 1:	SSI Control 0 (SSICR0), offset 0x000	704
Register 2:	SSI Control 1 (SSICR1), offset 0x004	706
Register 3:	SSI Data (SSIDR), offset 0x008	708
Register 4:	SSI Status (SSISR), offset 0x00C	709
Register 5:	SSI Clock Prescale (SSICPSR), offset 0x010	711
Register 6:	SSI Interrupt Mask (SSIIM), offset 0x014	712
Register 7:	SSI Raw Interrupt Status (SSIRIS), offset 0x018	713
Register 8:	SSI Masked Interrupt Status (SSIMIS), offset 0x01C	715
Register 9:	SSI Interrupt Clear (SSIICR), offset 0x020	717
Register 10:	SSI DMA Control (SSIDMACTL), offset 0x024	718
Register 11:	SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0	719
Register 12:	SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4	720
Register 13:	SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8	721
Register 14:	SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC	722
Register 15:	SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0	723
Register 16:	SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4	724
Register 17:	SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8	725
Register 18:	SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC	726
Register 19:	SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0	727
Register 20:	SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4	728
Register 21:	SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8	729
Register 22:	SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC	730
Inter-Integrated Circuit (I²C) Interface		731
Register 1:	I ² C Master Slave Address (I2CMSA), offset 0x000	747
Register 2:	I ² C Master Control/Status (I2CMCS), offset 0x004	748
Register 3:	I ² C Master Data (I2CMDR), offset 0x008	752
Register 4:	I ² C Master Timer Period (I2CMTPR), offset 0x00C	753
Register 5:	I ² C Master Interrupt Mask (I2CMIMR), offset 0x010	754
Register 6:	I ² C Master Raw Interrupt Status (I2CMRIS), offset 0x014	755
Register 7:	I ² C Master Masked Interrupt Status (I2CMMIS), offset 0x018	756
Register 8:	I ² C Master Interrupt Clear (I2CMICR), offset 0x01C	757
Register 9:	I ² C Master Configuration (I2CMCR), offset 0x020	758

Register 10:	I ² C Slave Own Address (I2CSOAR), offset 0x800	759
Register 11:	I ² C Slave Control/Status (I2CSCSR), offset 0x804	760
Register 12:	I ² C Slave Data (I2CSDR), offset 0x808	762
Register 13:	I ² C Slave Interrupt Mask (I2CSIMR), offset 0x80C	763
Register 14:	I ² C Slave Raw Interrupt Status (I2CSRIS), offset 0x810	764
Register 15:	I ² C Slave Masked Interrupt Status (I2CSMIS), offset 0x814	765
Register 16:	I ² C Slave Interrupt Clear (I2CSICR), offset 0x818	766
Inter-Integrated Circuit Sound (I²S) Interface		767
Register 1:	I ² S Transmit FIFO Data (I2STXFIFO), offset 0x000	780
Register 2:	I ² S Transmit FIFO Configuration (I2STXFIFOFCFG), offset 0x004	781
Register 3:	I ² S Transmit Module Configuration (I2STXCFG), offset 0x008	782
Register 4:	I ² S Transmit FIFO Limit (I2STXLIMIT), offset 0x00C	784
Register 5:	I ² S Transmit Interrupt Status and Mask (I2STXISM), offset 0x010	785
Register 6:	I ² S Transmit FIFO Level (I2STXLEV), offset 0x018	786
Register 7:	I ² S Receive FIFO Data (I2SRXFIFO), offset 0x800	787
Register 8:	I ² S Receive FIFO Configuration (I2SRXFIFOFCFG), offset 0x804	788
Register 9:	I ² S Receive Module Configuration (I2SRXCFG), offset 0x808	789
Register 10:	I ² S Receive FIFO Limit (I2SRXLIMIT), offset 0x80C	791
Register 11:	I ² S Receive Interrupt Status and Mask (I2SRXISM), offset 0x810	792
Register 12:	I ² S Receive FIFO Level (I2SRXLEV), offset 0x818	793
Register 13:	I ² S Module Configuration (I2SCFG), offset 0xC00	794
Register 14:	I ² S Interrupt Mask (I2SIM), offset 0xC10	796
Register 15:	I ² S Raw Interrupt Status (I2SRIS), offset 0xC14	798
Register 16:	I ² S Masked Interrupt Status (I2SMIS), offset 0xC18	800
Register 17:	I ² S Interrupt Clear (I2SIC), offset 0xC1C	802
Controller Area Network (CAN) Module		803
Register 1:	CAN Control (CANCTL), offset 0x000	824
Register 2:	CAN Status (CANSTS), offset 0x004	826
Register 3:	CAN Error Counter (CANERR), offset 0x008	829
Register 4:	CAN Bit Timing (CANBIT), offset 0x00C	830
Register 5:	CAN Interrupt (CANINT), offset 0x010	831
Register 6:	CAN Test (CANTST), offset 0x014	832
Register 7:	CAN Baud Rate Prescaler Extension (CANBRPE), offset 0x018	834
Register 8:	CAN IF1 Command Request (CANIF1CRQ), offset 0x020	835
Register 9:	CAN IF2 Command Request (CANIF2CRQ), offset 0x080	835
Register 10:	CAN IF1 Command Mask (CANIF1CMSK), offset 0x024	836
Register 11:	CAN IF2 Command Mask (CANIF2CMSK), offset 0x084	836
Register 12:	CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028	839
Register 13:	CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088	839
Register 14:	CAN IF1 Mask 2 (CANIF1MSK2), offset 0x02C	840
Register 15:	CAN IF2 Mask 2 (CANIF2MSK2), offset 0x08C	840
Register 16:	CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030	842
Register 17:	CAN IF2 Arbitration 1 (CANIF2ARB1), offset 0x090	842
Register 18:	CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034	843
Register 19:	CAN IF2 Arbitration 2 (CANIF2ARB2), offset 0x094	843
Register 20:	CAN IF1 Message Control (CANIF1MCTL), offset 0x038	845
Register 21:	CAN IF2 Message Control (CANIF2MCTL), offset 0x098	845

Register 22:	CAN IF1 Data A1 (CANIF1DA1), offset 0x03C	848
Register 23:	CAN IF1 Data A2 (CANIF1DA2), offset 0x040	848
Register 24:	CAN IF1 Data B1 (CANIF1DB1), offset 0x044	848
Register 25:	CAN IF1 Data B2 (CANIF1DB2), offset 0x048	848
Register 26:	CAN IF2 Data A1 (CANIF2DA1), offset 0x09C	848
Register 27:	CAN IF2 Data A2 (CANIF2DA2), offset 0x0A0	848
Register 28:	CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4	848
Register 29:	CAN IF2 Data B2 (CANIF2DB2), offset 0x0A8	848
Register 30:	CAN Transmission Request 1 (CANTXRQ1), offset 0x100	849
Register 31:	CAN Transmission Request 2 (CANTXRQ2), offset 0x104	849
Register 32:	CAN New Data 1 (CANNWDA1), offset 0x120	850
Register 33:	CAN New Data 2 (CANNWDA2), offset 0x124	850
Register 34:	CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140	851
Register 35:	CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144	851
Register 36:	CAN Message 1 Valid (CANMSG1VAL), offset 0x160	852
Register 37:	CAN Message 2 Valid (CANMSG2VAL), offset 0x164	852
Ethernet Controller		853
Register 1:	Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK), offset 0x000	868
Register 2:	Ethernet MAC Interrupt Mask (MACIM), offset 0x004	871
Register 3:	Ethernet MAC Receive Control (MACRCTL), offset 0x008	873
Register 4:	Ethernet MAC Transmit Control (MACTCTL), offset 0x00C	875
Register 5:	Ethernet MAC Data (MACDATA), offset 0x010	877
Register 6:	Ethernet MAC Individual Address 0 (MACIA0), offset 0x014	879
Register 7:	Ethernet MAC Individual Address 1 (MACIA1), offset 0x018	880
Register 8:	Ethernet MAC Threshold (MACTHR), offset 0x01C	881
Register 9:	Ethernet MAC Management Control (MACMCTL), offset 0x020	883
Register 10:	Ethernet MAC Management Divider (MACMDV), offset 0x024	885
Register 11:	Ethernet MAC Management Transmit Data (MACMTXD), offset 0x02C	886
Register 12:	Ethernet MAC Management Receive Data (MACMRXD), offset 0x030	887
Register 13:	Ethernet MAC Number of Packets (MACNP), offset 0x034	888
Register 14:	Ethernet MAC Transmission Request (MACTR), offset 0x038	889
Register 15:	Ethernet MAC Timer Support (MACTS), offset 0x03C	890
Register 16:	Ethernet MAC LED Encoding (MACLED), offset 0x040	891
Register 17:	Ethernet PHY MDIX (MDIX), offset 0x044	893
Register 18:	Ethernet PHY Management Register 0 – Control (MR0), address 0x00	894
Register 19:	Ethernet PHY Management Register 1 – Status (MR1), address 0x01	896
Register 20:	Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2), address 0x02	898
Register 21:	Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3), address 0x03	899
Register 22:	Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4), address 0x04	900
Register 23:	Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5), address 0x05	902
Register 24:	Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6), address 0x06	904
Register 25:	Ethernet PHY Management Register 16 – Vendor-Specific (MR16), address 0x10	905
Register 26:	Ethernet PHY Management Register 17 – Mode Control/Status (MR17), address 0x11	906
Register 27:	Ethernet PHY Management Register 27 – Special Control/Status (MR27), address 0x1B	908

Register 28:	Ethernet PHY Management Register 29 – Interrupt Status (MR29), address 0x1D	909
Register 29:	Ethernet PHY Management Register 30 – Interrupt Mask (MR30), address 0x1E	911
Register 30:	Ethernet PHY Management Register 31 – PHY Special Control/Status (MR31), address 0x1F	913
Universal Serial Bus (USB) Controller		914
Register 1:	USB Device Functional Address (USBFADDR), offset 0x000	942
Register 2:	USB Power (USBPOWER), offset 0x001	943
Register 3:	USB Transmit Interrupt Status (USBTXIS), offset 0x002	946
Register 4:	USB Receive Interrupt Status (USBRXIS), offset 0x004	948
Register 5:	USB Transmit Interrupt Enable (USBTXIE), offset 0x006	950
Register 6:	USB Receive Interrupt Enable (USBRXIE), offset 0x008	952
Register 7:	USB General Interrupt Status (USBIS), offset 0x00A	954
Register 8:	USB Interrupt Enable (USBIE), offset 0x00B	957
Register 9:	USB Frame Value (USBFRAME), offset 0x00C	960
Register 10:	USB Endpoint Index (USBEPIDX), offset 0x00E	961
Register 11:	USB Test Mode (USBTEST), offset 0x00F	962
Register 12:	USB FIFO Endpoint 0 (USBFIFO0), offset 0x020	964
Register 13:	USB FIFO Endpoint 1 (USBFIFO1), offset 0x024	964
Register 14:	USB FIFO Endpoint 2 (USBFIFO2), offset 0x028	964
Register 15:	USB FIFO Endpoint 3 (USBFIFO3), offset 0x02C	964
Register 16:	USB FIFO Endpoint 4 (USBFIFO4), offset 0x030	964
Register 17:	USB FIFO Endpoint 5 (USBFIFO5), offset 0x034	964
Register 18:	USB FIFO Endpoint 6 (USBFIFO6), offset 0x038	964
Register 19:	USB FIFO Endpoint 7 (USBFIFO7), offset 0x03C	964
Register 20:	USB FIFO Endpoint 8 (USBFIFO8), offset 0x040	964
Register 21:	USB FIFO Endpoint 9 (USBFIFO9), offset 0x044	964
Register 22:	USB FIFO Endpoint 10 (USBFIFO10), offset 0x048	964
Register 23:	USB FIFO Endpoint 11 (USBFIFO11), offset 0x04C	964
Register 24:	USB FIFO Endpoint 12 (USBFIFO12), offset 0x050	964
Register 25:	USB FIFO Endpoint 13 (USBFIFO13), offset 0x054	964
Register 26:	USB FIFO Endpoint 14 (USBFIFO14), offset 0x058	964
Register 27:	USB FIFO Endpoint 15 (USBFIFO15), offset 0x05C	964
Register 28:	USB Device Control (USBDEVCTL), offset 0x060	966
Register 29:	USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ), offset 0x062	968
Register 30:	USB Receive Dynamic FIFO Sizing (USBRXFIFOSZ), offset 0x063	968
Register 31:	USB Transmit FIFO Start Address (USBTXFIFOADD), offset 0x064	969
Register 32:	USB Receive FIFO Start Address (USBRXFIFOADD), offset 0x066	969
Register 33:	USB Connect Timing (USBCONTIM), offset 0x07A	970
Register 34:	USB OTG VBUS Pulse Timing (USBVPLEN), offset 0x07B	971
Register 35:	USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF), offset 0x07D	972
Register 36:	USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF), offset 0x07E	973
Register 37:	USB Transmit Functional Address Endpoint 0 (USBTXFUNCADDR0), offset 0x080	974
Register 38:	USB Transmit Functional Address Endpoint 1 (USBTXFUNCADDR1), offset 0x088	974
Register 39:	USB Transmit Functional Address Endpoint 2 (USBTXFUNCADDR2), offset 0x090	974
Register 40:	USB Transmit Functional Address Endpoint 3 (USBTXFUNCADDR3), offset 0x098	974
Register 41:	USB Transmit Functional Address Endpoint 4 (USBTXFUNCADDR4), offset 0x0A0	974
Register 42:	USB Transmit Functional Address Endpoint 5 (USBTXFUNCADDR5), offset 0x0A8	974
Register 43:	USB Transmit Functional Address Endpoint 6 (USBTXFUNCADDR6), offset 0x0B0	974

Register 44:	USB Transmit Functional Address Endpoint 7 (USBTXFUNCADDR7), offset 0x0B8	974
Register 45:	USB Transmit Functional Address Endpoint 8 (USBTXFUNCADDR8), offset 0x0C0	974
Register 46:	USB Transmit Functional Address Endpoint 9 (USBTXFUNCADDR9), offset 0x0C8	974
Register 47:	USB Transmit Functional Address Endpoint 10 (USBTXFUNCADDR10), offset 0x0D0	974
Register 48:	USB Transmit Functional Address Endpoint 11 (USBTXFUNCADDR11), offset 0x0D8	974
Register 49:	USB Transmit Functional Address Endpoint 12 (USBTXFUNCADDR12), offset 0x0E0	974
Register 50:	USB Transmit Functional Address Endpoint 13 (USBTXFUNCADDR13), offset 0x0E8	974
Register 51:	USB Transmit Functional Address Endpoint 14 (USBTXFUNCADDR14), offset 0x0F0	974
Register 52:	USB Transmit Functional Address Endpoint 15 (USBTXFUNCADDR15), offset 0x0F8	974
Register 53:	USB Transmit Hub Address Endpoint 0 (USBTXHUBADDR0), offset 0x082	976
Register 54:	USB Transmit Hub Address Endpoint 1 (USBTXHUBADDR1), offset 0x08A	976
Register 55:	USB Transmit Hub Address Endpoint 2 (USBTXHUBADDR2), offset 0x092	976
Register 56:	USB Transmit Hub Address Endpoint 3 (USBTXHUBADDR3), offset 0x09A	976
Register 57:	USB Transmit Hub Address Endpoint 4 (USBTXHUBADDR4), offset 0x0A2	976
Register 58:	USB Transmit Hub Address Endpoint 5 (USBTXHUBADDR5), offset 0x0AA	976
Register 59:	USB Transmit Hub Address Endpoint 6 (USBTXHUBADDR6), offset 0x0B2	976
Register 60:	USB Transmit Hub Address Endpoint 7 (USBTXHUBADDR7), offset 0x0BA	976
Register 61:	USB Transmit Hub Address Endpoint 8 (USBTXHUBADDR8), offset 0x0C2	976
Register 62:	USB Transmit Hub Address Endpoint 9 (USBTXHUBADDR9), offset 0x0CA	976
Register 63:	USB Transmit Hub Address Endpoint 10 (USBTXHUBADDR10), offset 0x0D2	976
Register 64:	USB Transmit Hub Address Endpoint 11 (USBTXHUBADDR11), offset 0x0DA	976
Register 65:	USB Transmit Hub Address Endpoint 12 (USBTXHUBADDR12), offset 0x0E2	976
Register 66:	USB Transmit Hub Address Endpoint 13 (USBTXHUBADDR13), offset 0x0EA	976
Register 67:	USB Transmit Hub Address Endpoint 14 (USBTXHUBADDR14), offset 0x0F2	976
Register 68:	USB Transmit Hub Address Endpoint 15 (USBTXHUBADDR15), offset 0x0FA	976
Register 69:	USB Transmit Hub Port Endpoint 0 (USBTXHUBPORT0), offset 0x083	978
Register 70:	USB Transmit Hub Port Endpoint 1 (USBTXHUBPORT1), offset 0x08B	978
Register 71:	USB Transmit Hub Port Endpoint 2 (USBTXHUBPORT2), offset 0x093	978
Register 72:	USB Transmit Hub Port Endpoint 3 (USBTXHUBPORT3), offset 0x09B	978
Register 73:	USB Transmit Hub Port Endpoint 4 (USBTXHUBPORT4), offset 0x0A3	978
Register 74:	USB Transmit Hub Port Endpoint 5 (USBTXHUBPORT5), offset 0x0AB	978
Register 75:	USB Transmit Hub Port Endpoint 6 (USBTXHUBPORT6), offset 0x0B3	978
Register 76:	USB Transmit Hub Port Endpoint 7 (USBTXHUBPORT7), offset 0x0BB	978
Register 77:	USB Transmit Hub Port Endpoint 8 (USBTXHUBPORT8), offset 0x0C3	978
Register 78:	USB Transmit Hub Port Endpoint 9 (USBTXHUBPORT9), offset 0x0CB	978
Register 79:	USB Transmit Hub Port Endpoint 10 (USBTXHUBPORT10), offset 0x0D3	978
Register 80:	USB Transmit Hub Port Endpoint 11 (USBTXHUBPORT11), offset 0x0DB	978
Register 81:	USB Transmit Hub Port Endpoint 12 (USBTXHUBPORT12), offset 0x0E3	978
Register 82:	USB Transmit Hub Port Endpoint 13 (USBTXHUBPORT13), offset 0x0EB	978
Register 83:	USB Transmit Hub Port Endpoint 14 (USBTXHUBPORT14), offset 0x0F3	978
Register 84:	USB Transmit Hub Port Endpoint 15 (USBTXHUBPORT15), offset 0x0FB	978
Register 85:	USB Receive Functional Address Endpoint 1 (USBRXFUNCADDR1), offset 0x08C	980
Register 86:	USB Receive Functional Address Endpoint 2 (USBRXFUNCADDR2), offset 0x094	980
Register 87:	USB Receive Functional Address Endpoint 3 (USBRXFUNCADDR3), offset 0x09C	980
Register 88:	USB Receive Functional Address Endpoint 4 (USBRXFUNCADDR4), offset 0x0A4	980
Register 89:	USB Receive Functional Address Endpoint 5 (USBRXFUNCADDR5), offset 0x0AC	980
Register 90:	USB Receive Functional Address Endpoint 6 (USBRXFUNCADDR6), offset 0x0B4	980
Register 91:	USB Receive Functional Address Endpoint 7 (USBRXFUNCADDR7), offset 0x0BC	980

Register 92:	USB Receive Functional Address Endpoint 8 (USBRXFUNCADDR8), offset 0x0C4	980
Register 93:	USB Receive Functional Address Endpoint 9 (USBRXFUNCADDR9), offset 0x0CC	980
Register 94:	USB Receive Functional Address Endpoint 10 (USBRXFUNCADDR10), offset 0x0D4	980
Register 95:	USB Receive Functional Address Endpoint 11 (USBRXFUNCADDR11), offset 0x0DC	980
Register 96:	USB Receive Functional Address Endpoint 12 (USBRXFUNCADDR12), offset 0x0E4	980
Register 97:	USB Receive Functional Address Endpoint 13 (USBRXFUNCADDR13), offset 0x0EC	980
Register 98:	USB Receive Functional Address Endpoint 14 (USBRXFUNCADDR14), offset 0x0F4	980
Register 99:	USB Receive Functional Address Endpoint 15 (USBRXFUNCADDR15), offset 0x0FC	980
Register 100:	USB Receive Hub Address Endpoint 1 (USBRXHUBADDR1), offset 0x08E	982
Register 101:	USB Receive Hub Address Endpoint 2 (USBRXHUBADDR2), offset 0x096	982
Register 102:	USB Receive Hub Address Endpoint 3 (USBRXHUBADDR3), offset 0x09E	982
Register 103:	USB Receive Hub Address Endpoint 4 (USBRXHUBADDR4), offset 0x0A6	982
Register 104:	USB Receive Hub Address Endpoint 5 (USBRXHUBADDR5), offset 0x0AE	982
Register 105:	USB Receive Hub Address Endpoint 6 (USBRXHUBADDR6), offset 0x0B6	982
Register 106:	USB Receive Hub Address Endpoint 7 (USBRXHUBADDR7), offset 0x0BE	982
Register 107:	USB Receive Hub Address Endpoint 8 (USBRXHUBADDR8), offset 0x0C6	982
Register 108:	USB Receive Hub Address Endpoint 9 (USBRXHUBADDR9), offset 0x0CE	982
Register 109:	USB Receive Hub Address Endpoint 10 (USBRXHUBADDR10), offset 0x0D6	982
Register 110:	USB Receive Hub Address Endpoint 11 (USBRXHUBADDR11), offset 0x0DE	982
Register 111:	USB Receive Hub Address Endpoint 12 (USBRXHUBADDR12), offset 0x0E6	982
Register 112:	USB Receive Hub Address Endpoint 13 (USBRXHUBADDR13), offset 0x0EE	982
Register 113:	USB Receive Hub Address Endpoint 14 (USBRXHUBADDR14), offset 0x0F6	982
Register 114:	USB Receive Hub Address Endpoint 15 (USBRXHUBADDR15), offset 0x0FE	982
Register 115:	USB Receive Hub Port Endpoint 1 (USBRXHUBPORT1), offset 0x08F	984
Register 116:	USB Receive Hub Port Endpoint 2 (USBRXHUBPORT2), offset 0x097	984
Register 117:	USB Receive Hub Port Endpoint 3 (USBRXHUBPORT3), offset 0x09F	984
Register 118:	USB Receive Hub Port Endpoint 4 (USBRXHUBPORT4), offset 0x0A7	984
Register 119:	USB Receive Hub Port Endpoint 5 (USBRXHUBPORT5), offset 0x0AF	984
Register 120:	USB Receive Hub Port Endpoint 6 (USBRXHUBPORT6), offset 0x0B7	984
Register 121:	USB Receive Hub Port Endpoint 7 (USBRXHUBPORT7), offset 0x0BF	984
Register 122:	USB Receive Hub Port Endpoint 8 (USBRXHUBPORT8), offset 0x0C7	984
Register 123:	USB Receive Hub Port Endpoint 9 (USBRXHUBPORT9), offset 0x0CF	984
Register 124:	USB Receive Hub Port Endpoint 10 (USBRXHUBPORT10), offset 0x0D7	984
Register 125:	USB Receive Hub Port Endpoint 11 (USBRXHUBPORT11), offset 0x0DF	984
Register 126:	USB Receive Hub Port Endpoint 12 (USBRXHUBPORT12), offset 0x0E7	984
Register 127:	USB Receive Hub Port Endpoint 13 (USBRXHUBPORT13), offset 0x0EF	984
Register 128:	USB Receive Hub Port Endpoint 14 (USBRXHUBPORT14), offset 0x0F7	984
Register 129:	USB Receive Hub Port Endpoint 15 (USBRXHUBPORT15), offset 0x0FF	984
Register 130:	USB Maximum Transmit Data Endpoint 1 (USBTXMAXP1), offset 0x110	986
Register 131:	USB Maximum Transmit Data Endpoint 2 (USBTXMAXP2), offset 0x120	986
Register 132:	USB Maximum Transmit Data Endpoint 3 (USBTXMAXP3), offset 0x130	986
Register 133:	USB Maximum Transmit Data Endpoint 4 (USBTXMAXP4), offset 0x140	986
Register 134:	USB Maximum Transmit Data Endpoint 5 (USBTXMAXP5), offset 0x150	986
Register 135:	USB Maximum Transmit Data Endpoint 6 (USBTXMAXP6), offset 0x160	986
Register 136:	USB Maximum Transmit Data Endpoint 7 (USBTXMAXP7), offset 0x170	986
Register 137:	USB Maximum Transmit Data Endpoint 8 (USBTXMAXP8), offset 0x180	986
Register 138:	USB Maximum Transmit Data Endpoint 9 (USBTXMAXP9), offset 0x190	986
Register 139:	USB Maximum Transmit Data Endpoint 10 (USBTXMAXP10), offset 0x1A0	986

Register 140: USB Maximum Transmit Data Endpoint 11 (USBTXMAXP11), offset 0x1B0	986
Register 141: USB Maximum Transmit Data Endpoint 12 (USBTXMAXP12), offset 0x1C0	986
Register 142: USB Maximum Transmit Data Endpoint 13 (USBTXMAXP13), offset 0x1D0	986
Register 143: USB Maximum Transmit Data Endpoint 14 (USBTXMAXP14), offset 0x1E0	986
Register 144: USB Maximum Transmit Data Endpoint 15 (USBTXMAXP15), offset 0x1F0	986
Register 145: USB Control and Status Endpoint 0 Low (USBCSRL0), offset 0x102	988
Register 146: USB Control and Status Endpoint 0 High (USBCSRH0), offset 0x103	992
Register 147: USB Receive Byte Count Endpoint 0 (USBCOUNT0), offset 0x108	994
Register 148: USB Type Endpoint 0 (USBTYPE0), offset 0x10A	995
Register 149: USB NAK Limit (USBNAKLMT), offset 0x10B	996
Register 150: USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1), offset 0x112	997
Register 151: USB Transmit Control and Status Endpoint 2 Low (USBTXCSRL2), offset 0x122	997
Register 152: USB Transmit Control and Status Endpoint 3 Low (USBTXCSRL3), offset 0x132	997
Register 153: USB Transmit Control and Status Endpoint 4 Low (USBTXCSRL4), offset 0x142	997
Register 154: USB Transmit Control and Status Endpoint 5 Low (USBTXCSRL5), offset 0x152	997
Register 155: USB Transmit Control and Status Endpoint 6 Low (USBTXCSRL6), offset 0x162	997
Register 156: USB Transmit Control and Status Endpoint 7 Low (USBTXCSRL7), offset 0x172	997
Register 157: USB Transmit Control and Status Endpoint 8 Low (USBTXCSRL8), offset 0x182	997
Register 158: USB Transmit Control and Status Endpoint 9 Low (USBTXCSRL9), offset 0x192	997
Register 159: USB Transmit Control and Status Endpoint 10 Low (USBTXCSRL10), offset 0x1A2	997
Register 160: USB Transmit Control and Status Endpoint 11 Low (USBTXCSRL11), offset 0x1B2	997
Register 161: USB Transmit Control and Status Endpoint 12 Low (USBTXCSRL12), offset 0x1C2	997
Register 162: USB Transmit Control and Status Endpoint 13 Low (USBTXCSRL13), offset 0x1D2	997
Register 163: USB Transmit Control and Status Endpoint 14 Low (USBTXCSRL14), offset 0x1E2	997
Register 164: USB Transmit Control and Status Endpoint 15 Low (USBTXCSRL15), offset 0x1F2	997
Register 165: USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1), offset 0x113	1002
Register 166: USB Transmit Control and Status Endpoint 2 High (USBTXCSRH2), offset 0x123	1002
Register 167: USB Transmit Control and Status Endpoint 3 High (USBTXCSRH3), offset 0x133	1002
Register 168: USB Transmit Control and Status Endpoint 4 High (USBTXCSRH4), offset 0x143	1002
Register 169: USB Transmit Control and Status Endpoint 5 High (USBTXCSRH5), offset 0x153	1002
Register 170: USB Transmit Control and Status Endpoint 6 High (USBTXCSRH6), offset 0x163	1002
Register 171: USB Transmit Control and Status Endpoint 7 High (USBTXCSRH7), offset 0x173	1002
Register 172: USB Transmit Control and Status Endpoint 8 High (USBTXCSRH8), offset 0x183	1002
Register 173: USB Transmit Control and Status Endpoint 9 High (USBTXCSRH9), offset 0x193	1002
Register 174: USB Transmit Control and Status Endpoint 10 High (USBTXCSRH10), offset 0x1A3	1002
Register 175: USB Transmit Control and Status Endpoint 11 High (USBTXCSRH11), offset 0x1B3	1002
Register 176: USB Transmit Control and Status Endpoint 12 High (USBTXCSRH12), offset 0x1C3	1002
Register 177: USB Transmit Control and Status Endpoint 13 High (USBTXCSRH13), offset 0x1D3	1002
Register 178: USB Transmit Control and Status Endpoint 14 High (USBTXCSRH14), offset 0x1E3	1002
Register 179: USB Transmit Control and Status Endpoint 15 High (USBTXCSRH15), offset 0x1F3	1002
Register 180: USB Maximum Receive Data Endpoint 1 (USBRXMAXP1), offset 0x114	1006
Register 181: USB Maximum Receive Data Endpoint 2 (USBRXMAXP2), offset 0x124	1006
Register 182: USB Maximum Receive Data Endpoint 3 (USBRXMAXP3), offset 0x134	1006
Register 183: USB Maximum Receive Data Endpoint 4 (USBRXMAXP4), offset 0x144	1006
Register 184: USB Maximum Receive Data Endpoint 5 (USBRXMAXP5), offset 0x154	1006
Register 185: USB Maximum Receive Data Endpoint 6 (USBRXMAXP6), offset 0x164	1006
Register 186: USB Maximum Receive Data Endpoint 7 (USBRXMAXP7), offset 0x174	1006
Register 187: USB Maximum Receive Data Endpoint 8 (USBRXMAXP8), offset 0x184	1006

Register 188: USB Maximum Receive Data Endpoint 9 (USBRXMAXP9), offset 0x194	1006
Register 189: USB Maximum Receive Data Endpoint 10 (USBRXMAXP10), offset 0x1A4	1006
Register 190: USB Maximum Receive Data Endpoint 11 (USBRXMAXP11), offset 0x1B4	1006
Register 191: USB Maximum Receive Data Endpoint 12 (USBRXMAXP12), offset 0x1C4	1006
Register 192: USB Maximum Receive Data Endpoint 13 (USBRXMAXP13), offset 0x1D4	1006
Register 193: USB Maximum Receive Data Endpoint 14 (USBRXMAXP14), offset 0x1E4	1006
Register 194: USB Maximum Receive Data Endpoint 15 (USBRXMAXP15), offset 0x1F4	1006
Register 195: USB Receive Control and Status Endpoint 1 Low (USBRXCSRL1), offset 0x116	1008
Register 196: USB Receive Control and Status Endpoint 2 Low (USBRXCSRL2), offset 0x126	1008
Register 197: USB Receive Control and Status Endpoint 3 Low (USBRXCSRL3), offset 0x136	1008
Register 198: USB Receive Control and Status Endpoint 4 Low (USBRXCSRL4), offset 0x146	1008
Register 199: USB Receive Control and Status Endpoint 5 Low (USBRXCSRL5), offset 0x156	1008
Register 200: USB Receive Control and Status Endpoint 6 Low (USBRXCSRL6), offset 0x166	1008
Register 201: USB Receive Control and Status Endpoint 7 Low (USBRXCSRL7), offset 0x176	1008
Register 202: USB Receive Control and Status Endpoint 8 Low (USBRXCSRL8), offset 0x186	1008
Register 203: USB Receive Control and Status Endpoint 9 Low (USBRXCSRL9), offset 0x196	1008
Register 204: USB Receive Control and Status Endpoint 10 Low (USBRXCSRL10), offset 0x1A6	1008
Register 205: USB Receive Control and Status Endpoint 11 Low (USBRXCSRL11), offset 0x1B6	1008
Register 206: USB Receive Control and Status Endpoint 12 Low (USBRXCSRL12), offset 0x1C6	1008
Register 207: USB Receive Control and Status Endpoint 13 Low (USBRXCSRL13), offset 0x1D6	1008
Register 208: USB Receive Control and Status Endpoint 14 Low (USBRXCSRL14), offset 0x1E6	1008
Register 209: USB Receive Control and Status Endpoint 15 Low (USBRXCSRL15), offset 0x1F6	1008
Register 210: USB Receive Control and Status Endpoint 1 High (USBRXCSRH1), offset 0x117	1013
Register 211: USB Receive Control and Status Endpoint 2 High (USBRXCSRH2), offset 0x127	1013
Register 212: USB Receive Control and Status Endpoint 3 High (USBRXCSRH3), offset 0x137	1013
Register 213: USB Receive Control and Status Endpoint 4 High (USBRXCSRH4), offset 0x147	1013
Register 214: USB Receive Control and Status Endpoint 5 High (USBRXCSRH5), offset 0x157	1013
Register 215: USB Receive Control and Status Endpoint 6 High (USBRXCSRH6), offset 0x167	1013
Register 216: USB Receive Control and Status Endpoint 7 High (USBRXCSRH7), offset 0x177	1013
Register 217: USB Receive Control and Status Endpoint 8 High (USBRXCSRH8), offset 0x187	1013
Register 218: USB Receive Control and Status Endpoint 9 High (USBRXCSRH9), offset 0x197	1013
Register 219: USB Receive Control and Status Endpoint 10 High (USBRXCSRH10), offset 0x1A7	1013
Register 220: USB Receive Control and Status Endpoint 11 High (USBRXCSRH11), offset 0x1B7	1013
Register 221: USB Receive Control and Status Endpoint 12 High (USBRXCSRH12), offset 0x1C7	1013
Register 222: USB Receive Control and Status Endpoint 13 High (USBRXCSRH13), offset 0x1D7	1013
Register 223: USB Receive Control and Status Endpoint 14 High (USBRXCSRH14), offset 0x1E7	1013
Register 224: USB Receive Control and Status Endpoint 15 High (USBRXCSRH15), offset 0x1F7	1013
Register 225: USB Receive Byte Count Endpoint 1 (USBRXCOUNT1), offset 0x118	1018
Register 226: USB Receive Byte Count Endpoint 2 (USBRXCOUNT2), offset 0x128	1018
Register 227: USB Receive Byte Count Endpoint 3 (USBRXCOUNT3), offset 0x138	1018
Register 228: USB Receive Byte Count Endpoint 4 (USBRXCOUNT4), offset 0x148	1018
Register 229: USB Receive Byte Count Endpoint 5 (USBRXCOUNT5), offset 0x158	1018
Register 230: USB Receive Byte Count Endpoint 6 (USBRXCOUNT6), offset 0x168	1018
Register 231: USB Receive Byte Count Endpoint 7 (USBRXCOUNT7), offset 0x178	1018
Register 232: USB Receive Byte Count Endpoint 8 (USBRXCOUNT8), offset 0x188	1018
Register 233: USB Receive Byte Count Endpoint 9 (USBRXCOUNT9), offset 0x198	1018
Register 234: USB Receive Byte Count Endpoint 10 (USBRXCOUNT10), offset 0x1A8	1018
Register 235: USB Receive Byte Count Endpoint 11 (USBRXCOUNT11), offset 0x1B8	1018

Register 236: USB Receive Byte Count Endpoint 12 (USBRXCOUNT12), offset 0x1C8	1018
Register 237: USB Receive Byte Count Endpoint 13 (USBRXCOUNT13), offset 0x1D8	1018
Register 238: USB Receive Byte Count Endpoint 14 (USBRXCOUNT14), offset 0x1E8	1018
Register 239: USB Receive Byte Count Endpoint 15 (USBRXCOUNT15), offset 0x1F8	1018
Register 240: USB Host Transmit Configure Type Endpoint 1 (USBTXTYPE1), offset 0x11A	1020
Register 241: USB Host Transmit Configure Type Endpoint 2 (USBTXTYPE2), offset 0x12A	1020
Register 242: USB Host Transmit Configure Type Endpoint 3 (USBTXTYPE3), offset 0x13A	1020
Register 243: USB Host Transmit Configure Type Endpoint 4 (USBTXTYPE4), offset 0x14A	1020
Register 244: USB Host Transmit Configure Type Endpoint 5 (USBTXTYPE5), offset 0x15A	1020
Register 245: USB Host Transmit Configure Type Endpoint 6 (USBTXTYPE6), offset 0x16A	1020
Register 246: USB Host Transmit Configure Type Endpoint 7 (USBTXTYPE7), offset 0x17A	1020
Register 247: USB Host Transmit Configure Type Endpoint 8 (USBTXTYPE8), offset 0x18A	1020
Register 248: USB Host Transmit Configure Type Endpoint 9 (USBTXTYPE9), offset 0x19A	1020
Register 249: USB Host Transmit Configure Type Endpoint 10 (USBTXTYPE10), offset 0x1AA	1020
Register 250: USB Host Transmit Configure Type Endpoint 11 (USBTXTYPE11), offset 0x1BA	1020
Register 251: USB Host Transmit Configure Type Endpoint 12 (USBTXTYPE12), offset 0x1CA	1020
Register 252: USB Host Transmit Configure Type Endpoint 13 (USBTXTYPE13), offset 0x1DA	1020
Register 253: USB Host Transmit Configure Type Endpoint 14 (USBTXTYPE14), offset 0x1EA	1020
Register 254: USB Host Transmit Configure Type Endpoint 15 (USBTXTYPE15), offset 0x1FA	1020
Register 255: USB Host Transmit Interval Endpoint 1 (USBTXINTERVAL1), offset 0x11B	1022
Register 256: USB Host Transmit Interval Endpoint 2 (USBTXINTERVAL2), offset 0x12B	1022
Register 257: USB Host Transmit Interval Endpoint 3 (USBTXINTERVAL3), offset 0x13B	1022
Register 258: USB Host Transmit Interval Endpoint 4 (USBTXINTERVAL4), offset 0x14B	1022
Register 259: USB Host Transmit Interval Endpoint 5 (USBTXINTERVAL5), offset 0x15B	1022
Register 260: USB Host Transmit Interval Endpoint 6 (USBTXINTERVAL6), offset 0x16B	1022
Register 261: USB Host Transmit Interval Endpoint 7 (USBTXINTERVAL7), offset 0x17B	1022
Register 262: USB Host Transmit Interval Endpoint 8 (USBTXINTERVAL8), offset 0x18B	1022
Register 263: USB Host Transmit Interval Endpoint 9 (USBTXINTERVAL9), offset 0x19B	1022
Register 264: USB Host Transmit Interval Endpoint 10 (USBTXINTERVAL10), offset 0x1AB	1022
Register 265: USB Host Transmit Interval Endpoint 11 (USBTXINTERVAL11), offset 0x1BB	1022
Register 266: USB Host Transmit Interval Endpoint 12 (USBTXINTERVAL12), offset 0x1CB	1022
Register 267: USB Host Transmit Interval Endpoint 13 (USBTXINTERVAL13), offset 0x1DB	1022
Register 268: USB Host Transmit Interval Endpoint 14 (USBTXINTERVAL14), offset 0x1EB	1022
Register 269: USB Host Transmit Interval Endpoint 15 (USBTXINTERVAL15), offset 0x1FB	1022
Register 270: USB Host Configure Receive Type Endpoint 1 (USBRXTYPE1), offset 0x11C	1024
Register 271: USB Host Configure Receive Type Endpoint 2 (USBRXTYPE2), offset 0x12C	1024
Register 272: USB Host Configure Receive Type Endpoint 3 (USBRXTYPE3), offset 0x13C	1024
Register 273: USB Host Configure Receive Type Endpoint 4 (USBRXTYPE4), offset 0x14C	1024
Register 274: USB Host Configure Receive Type Endpoint 5 (USBRXTYPE5), offset 0x15C	1024
Register 275: USB Host Configure Receive Type Endpoint 6 (USBRXTYPE6), offset 0x16C	1024
Register 276: USB Host Configure Receive Type Endpoint 7 (USBRXTYPE7), offset 0x17C	1024
Register 277: USB Host Configure Receive Type Endpoint 8 (USBRXTYPE8), offset 0x18C	1024
Register 278: USB Host Configure Receive Type Endpoint 9 (USBRXTYPE9), offset 0x19C	1024
Register 279: USB Host Configure Receive Type Endpoint 10 (USBRXTYPE10), offset 0x1AC	1024
Register 280: USB Host Configure Receive Type Endpoint 11 (USBRXTYPE11), offset 0x1BC	1024
Register 281: USB Host Configure Receive Type Endpoint 12 (USBRXTYPE12), offset 0x1CC	1024
Register 282: USB Host Configure Receive Type Endpoint 13 (USBRXTYPE13), offset 0x1DC	1024
Register 283: USB Host Configure Receive Type Endpoint 14 (USBRXTYPE14), offset 0x1EC	1024

Register 284: USB Host Configure Receive Type Endpoint 15 (USBRXTYPE15), offset 0x1FC	1024
Register 285: USB Host Receive Polling Interval Endpoint 1 (USBRXINTERVAL1), offset 0x11D	1026
Register 286: USB Host Receive Polling Interval Endpoint 2 (USBRXINTERVAL2), offset 0x12D	1026
Register 287: USB Host Receive Polling Interval Endpoint 3 (USBRXINTERVAL3), offset 0x13D	1026
Register 288: USB Host Receive Polling Interval Endpoint 4 (USBRXINTERVAL4), offset 0x14D	1026
Register 289: USB Host Receive Polling Interval Endpoint 5 (USBRXINTERVAL5), offset 0x15D	1026
Register 290: USB Host Receive Polling Interval Endpoint 6 (USBRXINTERVAL6), offset 0x16D	1026
Register 291: USB Host Receive Polling Interval Endpoint 7 (USBRXINTERVAL7), offset 0x17D	1026
Register 292: USB Host Receive Polling Interval Endpoint 8 (USBRXINTERVAL8), offset 0x18D	1026
Register 293: USB Host Receive Polling Interval Endpoint 9 (USBRXINTERVAL9), offset 0x19D	1026
Register 294: USB Host Receive Polling Interval Endpoint 10 (USBRXINTERVAL10), offset 0x1AD	1026
Register 295: USB Host Receive Polling Interval Endpoint 11 (USBRXINTERVAL11), offset 0x1BD	1026
Register 296: USB Host Receive Polling Interval Endpoint 12 (USBRXINTERVAL12), offset 0x1CD	1026
Register 297: USB Host Receive Polling Interval Endpoint 13 (USBRXINTERVAL13), offset 0x1DD	1026
Register 298: USB Host Receive Polling Interval Endpoint 14 (USBRXINTERVAL14), offset 0x1ED	1026
Register 299: USB Host Receive Polling Interval Endpoint 15 (USBRXINTERVAL15), offset 0x1FD	1026
Register 300: USB Request Packet Count in Block Transfer Endpoint 1 (USBRQPKTCOUNT1), offset 0x304	1028
Register 301: USB Request Packet Count in Block Transfer Endpoint 2 (USBRQPKTCOUNT2), offset 0x308	1028
Register 302: USB Request Packet Count in Block Transfer Endpoint 3 (USBRQPKTCOUNT3), offset 0x30C	1028
Register 303: USB Request Packet Count in Block Transfer Endpoint 4 (USBRQPKTCOUNT4), offset 0x310	1028
Register 304: USB Request Packet Count in Block Transfer Endpoint 5 (USBRQPKTCOUNT5), offset 0x314	1028
Register 305: USB Request Packet Count in Block Transfer Endpoint 6 (USBRQPKTCOUNT6), offset 0x318	1028
Register 306: USB Request Packet Count in Block Transfer Endpoint 7 (USBRQPKTCOUNT7), offset 0x31C	1028
Register 307: USB Request Packet Count in Block Transfer Endpoint 8 (USBRQPKTCOUNT8), offset 0x320	1028
Register 308: USB Request Packet Count in Block Transfer Endpoint 9 (USBRQPKTCOUNT9), offset 0x324	1028
Register 309: USB Request Packet Count in Block Transfer Endpoint 10 (USBRQPKTCOUNT10), offset 0x328	1028
Register 310: USB Request Packet Count in Block Transfer Endpoint 11 (USBRQPKTCOUNT11), offset 0x32C	1028
Register 311: USB Request Packet Count in Block Transfer Endpoint 12 (USBRQPKTCOUNT12), offset 0x330	1028
Register 312: USB Request Packet Count in Block Transfer Endpoint 13 (USBRQPKTCOUNT13), offset 0x334	1028
Register 313: USB Request Packet Count in Block Transfer Endpoint 14 (USBRQPKTCOUNT14), offset 0x338	1028
Register 314: USB Request Packet Count in Block Transfer Endpoint 15 (USBRQPKTCOUNT15), offset 0x33C	1028
Register 315: USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS), offset 0x340	1030
Register 316: USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS), offset 0x342	1032
Register 317: USB External Power Control (USBEPC), offset 0x400	1034

Register 318: USB External Power Control Raw Interrupt Status (USBEPCRIS), offset 0x404	1037
Register 319: USB External Power Control Interrupt Mask (USBEPCIM), offset 0x408	1038
Register 320: USB External Power Control Interrupt Status and Clear (USBEPCISC), offset 0x40C	1039
Register 321: USB Device RESUME Raw Interrupt Status (USBDRRIS), offset 0x410	1040
Register 322: USB Device RESUME Interrupt Mask (USBDRIM), offset 0x414	1041
Register 323: USB Device RESUME Interrupt Status and Clear (USBDRISC), offset 0x418	1042
Register 324: USB General-Purpose Control and Status (USBGPCS), offset 0x41C	1043
Register 325: USB VBUS Droop Control (USBVDC), offset 0x430	1044
Register 326: USB VBUS Droop Control Raw Interrupt Status (USBVDCRIS), offset 0x434	1045
Register 327: USB VBUS Droop Control Interrupt Mask (USBVDCIM), offset 0x438	1046
Register 328: USB VBUS Droop Control Interrupt Status and Clear (USBVDCISC), offset 0x43C	1047
Register 329: USB ID Valid Detect Raw Interrupt Status (USBIDVRIS), offset 0x444	1048
Register 330: USB ID Valid Detect Interrupt Mask (USBIDVIM), offset 0x448	1049
Register 331: USB ID Valid Detect Interrupt Status and Clear (USBIDVISC), offset 0x44C	1050
Register 332: USB DMA Select (USBDMASEL), offset 0x450	1051
Analog Comparators	1053
Register 1: Analog Comparator Masked Interrupt Status (ACMIS), offset 0x000	1058
Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004	1059
Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x008	1060
Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x010	1061
Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x020	1062
Register 6: Analog Comparator Status 1 (ACSTAT1), offset 0x040	1062
Register 7: Analog Comparator Control 0 (ACCTL0), offset 0x024	1063
Register 8: Analog Comparator Control 1 (ACCTL1), offset 0x044	1063
Pulse Width Modulator (PWM)	1065
Register 1: PWM Master Control (PWMCTL), offset 0x000	1079
Register 2: PWM Time Base Sync (PWMSYNC), offset 0x004	1080
Register 3: PWM Output Enable (PWMCENABLE), offset 0x008	1081
Register 4: PWM Output Inversion (PWMINVERT), offset 0x00C	1083
Register 5: PWM Output Fault (PWMAUTH), offset 0x010	1085
Register 6: PWM Interrupt Enable (PWMINTEN), offset 0x014	1087
Register 7: PWM Raw Interrupt Status (PWMRIS), offset 0x018	1089
Register 8: PWM Interrupt Status and Clear (PWMIISC), offset 0x01C	1091
Register 9: PWM Status (PWMSSTATUS), offset 0x020	1093
Register 10: PWM Fault Condition Value (PWMAUTHVAL), offset 0x024	1095
Register 11: PWM Enable Update (PWMCUPD), offset 0x028	1097
Register 12: PWM0 Control (PWMOCTL), offset 0x040	1100
Register 13: PWM1 Control (PWMICTL), offset 0x080	1100
Register 14: PWM2 Control (PWMCCTL), offset 0x0C0	1100
Register 15: PWM0 Interrupt and Trigger Enable (PWMOINTEN), offset 0x044	1105
Register 16: PWM1 Interrupt and Trigger Enable (PWMIINTEN), offset 0x084	1105
Register 17: PWM2 Interrupt and Trigger Enable (PWMCINTEN), offset 0x0C4	1105
Register 18: PWM0 Raw Interrupt Status (PWMORIS), offset 0x048	1108
Register 19: PWM1 Raw Interrupt Status (PWMI1RIS), offset 0x088	1108
Register 20: PWM2 Raw Interrupt Status (PWMC2RIS), offset 0x0C8	1108
Register 21: PWM0 Interrupt Status and Clear (PWMOISC), offset 0x04C	1110
Register 22: PWM1 Interrupt Status and Clear (PWMI1ISC), offset 0x08C	1110
Register 23: PWM2 Interrupt Status and Clear (PWMC2ISC), offset 0x0CC	1110

Register 24:	PWM0 Load (PWM0LOAD), offset 0x050	1112
Register 25:	PWM1 Load (PWM1LOAD), offset 0x090	1112
Register 26:	PWM2 Load (PWM2LOAD), offset 0x0D0	1112
Register 27:	PWM0 Counter (PWM0COUNT), offset 0x054	1113
Register 28:	PWM1 Counter (PWM1COUNT), offset 0x094	1113
Register 29:	PWM2 Counter (PWM2COUNT), offset 0x0D4	1113
Register 30:	PWM0 Compare A (PWM0CMPA), offset 0x058	1114
Register 31:	PWM1 Compare A (PWM1CMPA), offset 0x098	1114
Register 32:	PWM2 Compare A (PWM2CMPA), offset 0x0D8	1114
Register 33:	PWM0 Compare B (PWM0CMPB), offset 0x05C	1115
Register 34:	PWM1 Compare B (PWM1CMPB), offset 0x09C	1115
Register 35:	PWM2 Compare B (PWM2CMPB), offset 0x0DC	1115
Register 36:	PWM0 Generator A Control (PWM0GENA), offset 0x060	1116
Register 37:	PWM1 Generator A Control (PWM1GENA), offset 0x0A0	1116
Register 38:	PWM2 Generator A Control (PWM2GENA), offset 0xE0	1116
Register 39:	PWM0 Generator B Control (PWM0GENB), offset 0x064	1119
Register 40:	PWM1 Generator B Control (PWM1GENB), offset 0xA4	1119
Register 41:	PWM2 Generator B Control (PWM2GENB), offset 0xE4	1119
Register 42:	PWM0 Dead-Band Control (PWM0DBCTL), offset 0x068	1122
Register 43:	PWM1 Dead-Band Control (PWM1DBCTL), offset 0xA8	1122
Register 44:	PWM2 Dead-Band Control (PWM2DBCTL), offset 0xE8	1122
Register 45:	PWM0 Dead-Band Rising-Edge Delay (PWM0DBRISE), offset 0x06C	1123
Register 46:	PWM1 Dead-Band Rising-Edge Delay (PWM1DBRISE), offset 0x0AC	1123
Register 47:	PWM2 Dead-Band Rising-Edge Delay (PWM2DBRISE), offset 0x0EC	1123
Register 48:	PWM0 Dead-Band Falling-Edge-Delay (PWM0DBFALL), offset 0x070	1124
Register 49:	PWM1 Dead-Band Falling-Edge-Delay (PWM1DBFALL), offset 0xB0	1124
Register 50:	PWM2 Dead-Band Falling-Edge-Delay (PWM2DBFALL), offset 0xF0	1124
Register 51:	PWM0 Fault Source 0 (PWM0FLTSRC0), offset 0x074	1125
Register 52:	PWM1 Fault Source 0 (PWM1FLTSRC0), offset 0xB4	1125
Register 53:	PWM2 Fault Source 0 (PWM2FLTSRC0), offset 0xF4	1125
Register 54:	PWM0 Fault Source 1 (PWM0FLTSRC1), offset 0x078	1127
Register 55:	PWM1 Fault Source 1 (PWM1FLTSRC1), offset 0xB8	1127
Register 56:	PWM2 Fault Source 1 (PWM2FLTSRC1), offset 0xF8	1127
Register 57:	PWM0 Minimum Fault Period (PWM0MINFLTPER), offset 0x07C	1130
Register 58:	PWM1 Minimum Fault Period (PWM1MINFLTPER), offset 0x0BC	1130
Register 59:	PWM2 Minimum Fault Period (PWM2MINFLTPER), offset 0x0FC	1130
Register 60:	PWM0 Fault Pin Logic Sense (PWM0FLTSEN), offset 0x800	1131
Register 61:	PWM1 Fault Pin Logic Sense (PWM1FLTSEN), offset 0x880	1131
Register 62:	PWM2 Fault Pin Logic Sense (PWM2FLTSEN), offset 0x900	1131
Register 63:	PWM3 Fault Pin Logic Sense (PWM3FLTSEN), offset 0x980	1131
Register 64:	PWM0 Fault Status 0 (PWM0FLTSTAT0), offset 0x804	1132
Register 65:	PWM1 Fault Status 0 (PWM1FLTSTAT0), offset 0x884	1132
Register 66:	PWM2 Fault Status 0 (PWM2FLTSTAT0), offset 0x904	1132
Register 67:	PWM0 Fault Status 1 (PWM0FLTSTAT1), offset 0x808	1134
Register 68:	PWM1 Fault Status 1 (PWM1FLTSTAT1), offset 0x888	1134
Register 69:	PWM2 Fault Status 1 (PWM2FLTSTAT1), offset 0x908	1134
Quadrature Encoder Interface (QEI)	1137	
Register 1:	QEI Control (QEICTL), offset 0x000	1144

Register 2:	QEI Status (QEISTAT), offset 0x004	1147
Register 3:	QEI Position (QEIPOS), offset 0x008	1148
Register 4:	QEI Maximum Position (QEIMAXPOS), offset 0x00C	1149
Register 5:	QEI Timer Load (QEILOAD), offset 0x010	1150
Register 6:	QEI Timer (QEITIME), offset 0x014	1151
Register 7:	QEI Velocity Counter (QEICOUNT), offset 0x018	1152
Register 8:	QEI Velocity (QEISPEED), offset 0x01C	1153
Register 9:	QEI Interrupt Enable (QEINTEN), offset 0x020	1154
Register 10:	QEI Raw Interrupt Status (QEIRIS), offset 0x024	1156
Register 11:	QEI Interrupt Status and Clear (QEISC), offset 0x028	1158

Revision History

The revision history table notes changes made between the indicated revisions of the LM3S9L97 data sheet.

Table 1. Revision History

Date	Revision	Description
December 2010	8832	<ul style="list-style-type: none"> ■ Information on Advanced Encryption Standard (AES) cryptography tables and Cyclic Redundancy Check (CRC) error detection functionality was inadvertently omitted from some datasheets. This has been added. ■ In APINT register, changed bit name from SYSRESETREQ to SYSRESREQ. ■ Added DEBUG (Debug Priority) bit field to SYSPRI3 register. ■ Clarified Flash memory caution. ■ Restructured the General-Purpose Timer chapter to combine duplicated text. ■ Combined High and Low bit fields in GPTMTAILR, GPTMTAMATCHR, GPTMTAR, GPTMTAV, GPTMTBILR, GPTMTAMATCHR, GPTMTBR and GPTMTBV registers for compatibility with future releases. ■ Removed mention of false-start bit detection in the UART chapter. This feature is not supported. ■ Added SSI master clock restriction that SSIClk cannot be faster than 25 MHz. ■ Changed I²C master and slave register base addresses and offsets to be relative to I²C module base, so register base and offsets were changed for all I²C slave registers. ■ In Electrical Characteristics chapter: <ul style="list-style-type: none"> – Added single-ended clock source input voltage values to "Recommended DC Operating Conditions" table. – Deleted Oscillation mode value from "MOSC Oscillator Input Characteristics" table. – Added T_{VDD2_3} supply voltage parameter to "Reset Characteristics" table. – Added "Power-On Reset and Voltage Parameters" timing diagram. – Added $t_{VDDRISE_HIB}$ supply voltage parameter to "Hibernation Module AC Characteristics" table. – Added "VDD Ramp when Waking from Hibernation" timing diagram.

Table 1. Revision History (continued)

Date	Revision	Description
September 2010	7794	<ul style="list-style-type: none"> ■ Reorganized ARM Cortex-M3 Processor Core, Memory Map and Interrupts chapters, creating two new chapters, The Cortex-M3 Processor and Cortex-M3 Peripherals. Much additional content was added, including all the Cortex-M3 registers. ■ Changed register names to be consistent with StellarisWare® names: the Cortex-M3 Interrupt Control and Status (ICSR) register to the Interrupt Control and State (INTCTRL) register, and the Cortex-M3 Interrupt Set Enable (SETNA) register to the Interrupt 0-31 Set Enable (EN0) register. ■ In the System Control chapter: <ul style="list-style-type: none"> – Corrected Reset Sources table (see Table 5-3 on page 195). – Added section “Special Considerations for Reset when Debugging” on page 199. ■ In the Hibernation Module chapter, added section “Special Considerations When Using a 4.194304-MHz Crystal” on page 306. ■ Clarified how reset operation affects the Hibernation module (“Register Reset” on page 310). ■ In the Internal Memory chapter: <ul style="list-style-type: none"> – Added clarification of instruction execution during Flash operations. – Deleted ROM Version (RMVER) register as it is not used. ■ Modified Figure 9-1 on page 428 and Figure 9-2 on page 429 to clarify operation of the GPIO inputs when used as an alternate function. ■ Corrected GPIOAMSEL bit field in GPIO Analog Mode Select (GPIOAMSEL) register to be eight-bits wide, bits[7:0]. ■ In General-Purpose Timers chapter, clarified operation of the 32-bit RTC mode. ■ In CAN chapter, clarified CAN bit timing examples. ■ In Operating Characteristics chapter, corrected Thermal resistance (junction to ambient) value to 32. ■ In Electrical Characteristics chapter: <ul style="list-style-type: none"> – Added “Input voltage for a GPIO configured as an analog input” value to Table 26-1 on page 1232. – Added I_{LKG} parameter (GPIO input leakage current) to Table 26-6 on page 1234. – Corrected Nom values for I_{HIB_NORTC} and I_{HIB_RTC} in Table 26-9 on page 1234. – Corrected reset timing in Table 26-20 on page 1239. – Corrected values for $t_{WAKE_TO_HIB}$ in Table 26-22 on page 1242. – Specified Max value for V_{REFA} in Table 26-25 on page 1244. – Corrected values for t_{CLKRF} (SSIClk rise/fall time) in Table 26-27 on page 1245. – Added I²C Characteristics table (see Table 26-28 on page 1246). ■ Added dimensions for Tray and Tape and Reel shipping mediums.
June 2010	7413	<ul style="list-style-type: none"> ■ In “Thermal Characteristics” table, corrected thermal resistance value from 34 to 32.

Table 1. Revision History (continued)

Date	Revision	Description
June 2010	7299	<ul style="list-style-type: none"> ■ Removed 4.194304-MHz crystal as a source for the system clock and PLL. ■ Summarized ROM contents descriptions in the "Internal Memory" chapter and removed various ROM appendices. ■ Clarified DMA channel terminology: changed name of DMA Channel Alternate Select (DMACHALT) register to DMA Channel Assignment (DMACHASGN) register, changed CHALT bit field to CHASGN, and changed terminology from primary and alternate channels to primary and secondary channels. ■ In Signal Tables chapter, added table "Connections for Unused Signals." ■ In "Electrical Characteristics" chapter: <ul style="list-style-type: none"> – In "Reset Characteristics" table, corrected Supply voltage (VDD) rise time. – Clarified figure "SDRAM Initialization and Load Mode Register Timing".
May 2010	7164	<ul style="list-style-type: none"> ■ Added data sheets for five new Stellaris® Tempest-class parts: LM3S1R26, LM3S1621, LM3S1B21, LM3S9781, and LM3S9B81. ■ Additional minor data sheet clarifications and corrections.
May 2010	7101	<ul style="list-style-type: none"> ■ Added pin table "Possible Pin Assignments for Alternate Functions", which lists the signals based on number of possible pin assignments. This table can be used to plan how to configure the pins for a particular functionality. ■ Additional minor data sheet clarifications and corrections.
March 2010	6983	<ul style="list-style-type: none"> ■ Extended TBRL bit field in GPTMTBR register. ■ Additional minor data sheet clarifications and corrections.
March 2010	6912	<ul style="list-style-type: none"> ■ Renamed the USER_DBG register to the BOOTCFG register in the Internal Memory chapter. Added information on how to use a GPIO pin to force the ROM Boot Loader to execute on reset. ■ Added three figures to the ADC chapter on sample phase control. ■ Clarified configuration of USB0VBUS and USB0ID in OTG mode.

Table 1. Revision History (continued)

Date	Revision	Description
February 2010	6790	<ul style="list-style-type: none"> ■ Added 108-ball BGA package. ■ In "System Control" chapter: <ul style="list-style-type: none"> – Clarified functional description for external reset and brown-out reset. – Clarified Debug Access Port operation after Sleep modes. – Corrected the reset value of the Run-Mode Clock Configuration 2 (RCC2) register. ■ In "Internal Memory" chapter, clarified wording on Flash memory access errors and added a section on interrupts to the Flash memory description. ■ Added clarification about timer operating modes and added register descriptions for the GPTM Timer n Prescale Match (GPTMTnPMR) registers. ■ Clarified register descriptions for GPTM Timer A Value (GPTMTAV) and GPTM Timer B Value (GPTMTBV) registers. ■ Corrected the reset value of the ADC Sample Sequence Result FIFO n (ADCSSFIFO_n) registers. ■ Added ADC Sample Phase Control (ADCSPC) register at offset 0x24. ■ Added caution note to the I²C Master Timer Period (I2CMTPR) register description and changed field width to 7 bits. ■ In the "Controller Area Network" chapter, added clarification about reading from the CAN FIFO buffer and clarified packet timestamps functional description. ■ In the "Ethernet Controller" chapter: <ul style="list-style-type: none"> – Corrected the reset value and the LED1 bit positions of the Ethernet MAC LED Encoding (MACLED) register. – Added clarification about the use of the NPR field in the Ethernet MAC Number of Packets (MACNP) register. – Corrected reset values for Ethernet PHY Management Register 0 – Control (MR0) and Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5) registers. ■ Added Session Disconnect (DISCON) bit to the USB General Interrupt Status (USBIS) and USB Interrupt Enable (USBIE) registers. ■ Made these changes to the Operating Characteristics chapter: <ul style="list-style-type: none"> – Added storage temperature ratings to "Temperature Characteristics" table – Added "ESD Absolute Maximum Ratings" table ■ Made these changes to the Electrical Characteristics chapter: <ul style="list-style-type: none"> – In "Flash Memory Characteristics" table, corrected Mass erase time – Added sleep and deep-sleep wake-up times ("Sleep Modes AC Characteristics" table) – In "Reset Characteristics" table, corrected units for supply voltage (VDD) rise time – Modified the preliminary current consumption specification for Run mode 1 and Deep-Sleep mode. – Added table entry for VDD3ON power consumption to Table 26-9 on page 1234. ■ Added additional DriverLib functions to appendix.

Table 1. Revision History (continued)

Date	Revision	Description
October 2009	6458	<ul style="list-style-type: none"> ■ Released new 1000, 3000, 5000 and 9000 series Stellaris® devices. ■ The IDCODE value was corrected to be 0x4BA0.0477. ■ Clarified that the NMISSET bit in the ICSR register in the NVIC is also a source for NMI. ■ Clarified the use of the LDO. ■ To clarify clock operation, reorganized clocking section, changed the USEFRACt bit to the DIV400 bit and the FRACt bit to the SYSDIV2LSB bit in the RCC2 register, added tables, and rewrote descriptions. ■ Corrected bit description of the DSDIVORIDE field in the DSLPCLKCFG register. ■ Removed the DSFLASHCFG register at System Control offset 0x14C as it does not function correctly. ■ Removed the MAXADC1SPD and MAXADC0SPD fields from the DCGC0 as they have no function in deep-sleep mode. ■ Corrected address offsets for the Flash Write Buffer (FWBn) registers. ■ Added Flash Control (FCTL) register at Internal memory offset 0x0F8 to help control frequent power cycling when hibernation is not used. ■ Changed the name of the EPI channels for clarification: EPI0_TX became EPI0_WFIFO and EPI0_RX became EPI0_NBRFIFO. This change was also made in the DC7 bit descriptions. ■ Removed the DMACHIS register at DMA module offset 0x504 as it does not function correctly. ■ Corrected alternate channel assignments for the μDMA controller. ■ Major improvements to the EPI chapter. ■ EPISDRAMCFG2 register was deleted as its function is not needed. ■ Clarified CAN bit timing and corrected examples. ■ Added pseudo-code for MDI/MDIX operation. ■ Corrected reset value of the MR1 register to 0x7809. ■ Clarified PWM source for ADC triggering ■ Corrected ADDR field in the USBTXFIFOADD register to be 9 bits instead of 13 bits. ■ Changed SSI set up and hold times to be expressed in system clocks, not ns. ■ Updated Electrical Characteristics chapter with latest data. Changes were made to Hibernation, ADC and EPI content. ■ Additional minor data sheet clarifications and corrections.

Table 1. Revision History (continued)

Date	Revision	Description
July 2009	5930	<ul style="list-style-type: none"> ■ Corrected values for MAXADC0SPD and MAXADC1SPD bits in DC1, RCGC0, SCGC0, and DCGC0 registers. ■ Corrected figure "TI Synchronous Serial Frame Format (Single Transfer)". ■ Added description for Ethernet PHY power-saving modes. ■ Changed HIB pin from type TTL to type OD. ■ Made a number of corrections to the Electrical Characteristics chapter: <ul style="list-style-type: none"> – Deleted V_{BAT} and V_{REFA} parameters from and added footnotes to Recommended DC Operating Conditions table. – Modified Hibernation Module DC Characteristics table. – Deleted Nominal and Maximum Current Specifications section. – Deleted SDRAM Read Command Timing, SDRAM Write Command Timing, SDRAM Write Burst Timing, SDRAM Precharge Command Timing and SDRAM CAS Latency Timing figures and replaced with SDRAM Read Timing and SDRAM Write Timing figures. – Modified Host-Bus 8/16 Mode Write Timing figure. – Modified General-Purpose Mode Read and Write Timing figure. – Major changes to ADC Characteristics tables, including adding additional tables and diagram. ■ Added missing ROM_I2SIntStatus function to ROM DriverLib Functions appendix. ■ Corrected ordering part numbers. ■ Additional minor data sheet clarifications and corrections.
June 2009	5779	<ul style="list-style-type: none"> ■ In System Control chapter, clarified power-on reset and external reset pin descriptions in "Reset Sources" section. ■ Added missing comparator output pin bits to DC3 register; reset value changed as well. ■ Clarified explanation of nonvolatile register programming in Internal Memory chapter. ■ Added explanation of reset value to FMPRE0/1/2/3, FMPPE0/1/2/3, USER_DBG, and USER_REG0 registers. ■ In Request Type Support table in DMA chapter, corrected general-purpose timer row. ■ In General-Purpose Timers chapter, clarified DMA operation. ■ Added table "Preliminary Current Consumption" to Characteristics chapter. ■ Corrected Nom and Max values in "Hibernation Detailed Current Specifications" table. ■ Corrected Nom and Max values in EPI Characteristics table. ■ Added "CSn to output invalid" parameter to EPI table "EPI Host-Bus 8 and Host-Bus 16 Interface Characteristics" and figure "Host-Bus 8/16 Mode Read Timing". ■ Corrected INL, DNL, OFF and GAIN values in ADC Characteristics table. ■ Updated ROM DriverLib appendix with RevC0 functions. ■ Updated part ordering numbers. ■ Additional minor data sheet clarifications and corrections.

Table 1. Revision History (*continued*)

Date	Revision	Description
May 2009	5285	Started tracking revision history.

About This Document

This data sheet provides reference information for the LM3S9L97 microcontroller, describing the functional blocks of the system-on-chip (SoC) device designed around the ARM® Cortex™-M3 core.

Audience

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

About This Manual

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

Related Documents

The following related documents are available on the Stellaris® web site at www.ti.com/stellaris:

- *Stellaris® Errata*
- *ARM® Cortex™-M3 Errata*
- *Cortex™-M3 Instruction Set Technical User's Manual*
- *Stellaris® Boot Loader User's Guide*
- *Stellaris® Graphics Library User's Guide*
- *Stellaris® Peripheral Driver Library User's Guide*
- *Stellaris® ROM User's Guide*
- *Stellaris® USB Library User's Guide*

The following related documents are also referenced:

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

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

Documentation Conventions

This document uses the conventions shown in Table 2 on page 47.

Table 2. Documentation Conventions

Notation	Meaning
General Register Notation	
REGISTER	APB registers are indicated in uppercase bold. For example, PBORCTL 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, SRCRn represents any (or all) of the three Software Reset Control registers: SRCR0 , SRCR1 , and SRCR2 .
bit	A single bit in a register.
bit field	Two or more consecutive and related bits.
offset 0xnnnn	A hexadecimal increment to a register's address, relative to that module's base address as specified in Table 2-4 on page 92.
Register N	Registers are numbered consecutively throughout the document to aid in referencing them. The register number has no meaning to software.
reserved	Register bits marked <i>reserved</i> are reserved for future use. In most cases, reserved bits are set to 0; however, user software should not rely on the value of a reserved bit. To provide software compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
yy:xx	The range of register bits inclusive from xx to yy. For example, 31:15 means bits 15 through 31 in that register.
Register Bit/Field Types	This value in the register bit diagram indicates whether software running on the controller can change the value of the bit field.
RC	Software can read this field. The bit or field is cleared by hardware after reading the bit/field.
RO	Software can read this field. Always write the chip reset value.
R/W	Software can read or write this field.
R/WC	Software can read or write this field. Writing to it with any value clears the register.
R/W1C	Software can read or write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged. This register type is primarily used for clearing interrupt status bits where the read operation provides the interrupt status and the write of the read value clears only the interrupts being reported at the time the register was read.
R/W1S	Software can read or write a 1 to this field. A write of a 0 to a R/W1S bit does not affect the bit value in the register.
W1C	Software can write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged. A read of the register returns no meaningful data. This register is typically used to clear the corresponding bit in an interrupt register.
WO	Only a write by software is valid; a read of the register returns no meaningful data.
Register Bit/Field Reset Value	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.
Pin/Signal Notation	
[]	Pin alternate function; a pin defaults to the signal without the brackets.
pin	Refers to the physical connection on the package.
signal	Refers to the electrical signal encoding of a pin.

Table 2. Documentation Conventions (*continued*)

Notation	Meaning
assert a signal	Change the value of the signal from the logically False state to the logically True state. For active High signals, the asserted signal value is 1 (High); for active Low signals, the asserted signal value is 0 (Low). The active polarity (High or Low) is defined by the signal name (see SIGNAL and <u>SIGNAL</u> below).
deassert a signal	Change the value of the signal from the logically True state to the logically False state.
<u>SIGNAL</u>	Signal names are in uppercase and in the Courier font. An overbar on a signal name indicates that it is active Low. To assert <u>SIGNAL</u> is to drive it Low; to deassert <u>SIGNAL</u> is to drive it High.
SIGNAL	Signal names are in uppercase and in the Courier font. An active High signal has no overbar. To assert SIGNAL is to drive it High; to deassert SIGNAL is to drive it Low.
Numbers	
X	An uppercase X indicates any of several values is allowed, where X can be any legal pattern. For example, a binary value of 0X00 can be either 0100 or 0000, a hex value of 0xX is 0x0 or 0x1, and so on.
0x	Hexadecimal numbers have a prefix of 0x. For example, 0x0FF is the hexadecimal number FF. All other numbers within register tables are assumed to be binary. Within conceptual information, binary numbers are indicated with a b suffix, for example, 1011b, and decimal numbers are written without a prefix or suffix.

1 Architectural Overview

Texas Instruments 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 Texas Instruments 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 LM3S9L97 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 memory up to 50 MHz; a prefetch buffer improves performance above 50 MHz
 - 48 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
- Advanced Serial Integration
 - 10/100 Ethernet MAC and PHY with IEEE 1588 PTP hardware support
 - 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 I²C modules
 - Two Synchronous Serial Interface modules (SSI)

- Integrated Interchip Sound (I²S) 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)
 - Lower-power battery-backed hibernation module
 - Real-Time Clock in Hibernation module
 - Two Watchdog Timers
 - One timer runs off the main oscillator
 - One timer runs off the precision internal oscillator
 - Up to 60 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
 - Six 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 16 analog input channels and sample rate of one million samples/second
 - Two analog comparators
 - Eight digital comparators
 - On-chip voltage regulator
- JTAG and ARM Serial Wire Debug (SWD)
- 100-pin LQFP and 108-ball BGA package
- Industrial (-40°C to 85°C) Temperature Range

The LM3S9L97 microcontroller is targeted for industrial applications, including remote monitoring, electronic point-of-sale machines, test and measurement equipment, network appliances and switches, factory automation, HVAC and building control, gaming equipment, motion control, medical instrumentation, and fire and security.

For applications requiring extreme conservation of power, the LM3S9L97 microcontroller features a battery-backed Hibernation module to efficiently power down the LM3S9L97 to a low-power state during extended periods of inactivity. With a power-up/power-down sequencer, a continuous time counter (RTC), a pair of match registers, an APB interface to the system bus, and dedicated non-volatile memory, the Hibernation module positions the LM3S9L97 microcontroller perfectly for battery applications.

In addition, the LM3S9L97 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 LM3S9L97 microcontroller is code-compatible to all members of the extensive Stellaris family; providing flexibility to fit our customers' precise needs.

Texas Instruments offers a complete solution to get to market quickly, with evaluation and development boards, white papers and application notes, an easy-to-use peripheral driver library, and a strong support, sales, and distributor network. See "Ordering and Contact Information" on page 1304 for ordering information for Stellaris family devices.

1.1 Functional Overview

The following sections provide an overview of the features of the LM3S9L97 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 1304.

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 73)

All members of the Stellaris product family, including the LM3S9L97 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 architecture optimized for small-footprint embedded applications
- Outstanding processing performance combined with fast interrupt handling
- Thumb-2 mixed 16-/32-bit instruction set delivers the high performance expected of 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
- Fast code execution permits slower processor clock or increases sleep mode time
- Harvard architecture characterized by separate buses for instruction and data

- Efficient processor core, system and memories
- Hardware division and fast multiplier
- Deterministic, high-performance interrupt handling for time-critical applications
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionality
- Enhanced system debug with extensive breakpoint and trace capabilities
- Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging and tracing
- Migration from the ARM7™ processor family for better performance and power efficiency
- Optimized for single-cycle Flash memory usage
- Ultra-low power consumption with integrated sleep modes
- 80-MHz operation
- 1.25 DMIPS/MHz

1.1.1.2 **Memory Map (see page 92)**

A memory map lists the location of instructions and data in memory. The memory map for the LM3S9L97 controller can be found in “Memory Model” on page 92. Register addresses are given as a hexadecimal increment, relative to the module’s base address as shown in the memory map.

1.1.1.3 **System Timer (SysTick) (see page 116)**

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

1.1.1.4 **Nested Vectored Interrupt Controller (NVIC) (see page 117)**

The LM3S9L97 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 51 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

1.1.1.5 System Control Block (SCB) (see page 119)

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

1.1.1.6 Memory Protection Unit (MPU) (see page 119)

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

1.1.2 On-Chip Memory

The following sections describe the on-chip memory modules.

1.1.2.1 SRAM (see page 329)

The LM3S9L97 microcontroller provides 48 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 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 (μ DMA).

1.1.2.2 Flash Memory (see page 331)

The LM3S9L97 microcontroller provides 128 KB of single-cycle on-chip Flash memory (above 50 MHz, the Flash memory can be accessed in a single cycle as long as the code is linear; branches incur a one-cycle stall). The Flash memory is organized as a set of 1-KB blocks that can be individually erased. Erasing a block causes the entire contents of the block to be reset to all 1s. These blocks are paired into a set of 2-KB blocks that can be individually protected. The blocks can be marked as read-only or execute-only, providing different levels of code protection. Read-only blocks cannot be erased or programmed, protecting the contents of those blocks from being modified. Execute-only blocks cannot be erased or programmed, and can only be read by the controller instruction fetch mechanism, protecting the contents of those blocks from being read by either the controller or by a debugger.

1.1.2.3 ROM (see page 329)

The LM3S9L97 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

The Stellaris Peripheral Driver Library is 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 Texas Instruments 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 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.

1.1.3 Serial Communications Peripherals

The LM3S9L97 controller supports both asynchronous and synchronous serial communications with:

- 10/100 Ethernet MAC and PHY with IEEE 1588 PTP hardware support
- 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 I²C modules
- Two Synchronous Serial Interface modules (SSI)
- Integrated Interchip Sound (I²S) Module

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

1.1.3.1 Ethernet Controller (see page 853)

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

The Stellaris Ethernet Controller consists of a fully integrated media access controller (MAC) and network physical (PHY) interface and has the following features:

- Conforms to the *IEEE 802.3-2002 specification*
 - 10BASE-T/100BASE-TX IEEE-802.3 compliant. Requires only a dual 1:1 isolation transformer interface to the line
 - 10BASE-T/100BASE-TX ENDEC, 100BASE-TX scrambler/descrambler
 - Full-featured auto-negotiation
- Multiple operational modes
 - Full- and half-duplex 100 Mbps
 - Full- and half-duplex 10 Mbps
 - Power-saving and power-down modes
- Highly configurable
 - Programmable MAC address
 - LED activity selection
 - Promiscuous mode support
 - CRC error-rejection control
 - User-configurable interrupts
- Physical media manipulation
 - MDI/MDI-X cross-over support through software assist
 - Register-programmable transmit amplitude
 - Automatic polarity correction and 10BASE-T signal reception
- IEEE 1588 Precision Time Protocol: Provides highly accurate time stamps for individual packets
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Separate channels for transmit and receive
 - Receive channel request asserted on packet receipt
 - Transmit channel request asserted on empty transmit FIFO

1.1.3.2 Controller Area Network (see page 803)

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 1 Mbps 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 LM3S9L97 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

1.1.3.3 USB (see page 914)

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 LM3S9L97 microcontroller 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
- 32 endpoints
 - 1 dedicated control IN endpoint and 1 dedicated control OUT endpoint
 - 15 configurable IN endpoints and 15 configurable OUT endpoints
- 4 KB dedicated endpoint memory: one endpoint may be defined for double-buffered 1023-byte isochronous packet size
- VBUS droop and valid ID detection and interrupt
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)

- Separate channels for transmit and receive for up to three IN endpoints and three OUT endpoints
- Channel requests asserted when FIFO contains required amount of data

1.1.3.4 UART (see page 627)

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 LM3S9L97 microcontroller 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 three 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
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
- Standard asynchronous communication bits for start, stop, and parity
- Line-break generation and detection
- Fully programmable serial interface characteristics
 - 5, 6, 7, or 8 data bits
 - Even, odd, stick, or no-parity bit generation/detection
 - 1 or 2 stop bit generation
- 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 μ 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)
- LIN protocol support

- Standard FIFO-level and End-of-Transmission interrupts
- Efficient transfers using Micro Direct Memory Access Controller (μ 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

1.1.3.5 I²C (see page 731)

The Inter-Integrated Circuit (I²C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL). The I²C bus interfaces to external I²C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I²C bus may also be used for system testing and diagnostic purposes in product development and manufacture.

Each device on the I²C bus can be designated as either a master or a slave. Each I²C 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 I²C master and slave can generate interrupts.

The LM3S9L97 microcontroller includes two I²C modules with the following features:

- Devices on the I²C bus can be designated as either a master or a slave
 - Supports both transmitting and receiving data as either a master or a slave
 - Supports simultaneous master and slave operation
- Four I²C 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 transferred 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

1.1.3.6 SSI (see page 688)

Synchronous Serial Interface (SSI) is a four-wire bi-directional communications interface that converts data between parallel and serial. The SSI module performs serial-to-parallel conversion on data received from a peripheral device, and parallel-to-serial conversion on data transmitted to a peripheral device. The 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.

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

The LM3S9L97 microcontroller includes two SSI modules with 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, each 16 bits wide and 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 (μ 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

1.1.3.7 Inter-Integrated Circuit Sound (I²S) Interface (see page 767)

The I²S interface is a configurable serial audio core that contains a transmit module and a receive module. The module is configurable for the I²S 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 I²S interface has the following features:

- Configurable audio format supporting I²S, 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 μDMA 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 (μDMA)
 - Separate channels for transmit and receive
 - Burst requests
 - Channel requests asserted when FIFO contains required amount of data

1.1.4 System Integration

The LM3S9L97 microcontroller provides a variety of standard system functions integrated into the device, including:

- Micro Direct Memory Access Controller (μDMA)
- 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)
- Lower-power battery-backed hibernation module
- Real-Time Clock in Hibernation module
- Two Watchdog Timers
- Up to 60 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.

1.1.4.1 Direct Memory Access (see page 365)

The LM3S9L97 microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (μ DMA). The μ 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 available bus bandwidth. The μ 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 μ DMA controller provides the following features:

- ARM PrimeCell® 32-channel configurable μ DMA controller
- Support for memory-to-memory, memory-to-peripheral, and peripheral-to-memory in multiple transfer modes
 - 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
 - Primary and secondary 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 μ DMA controller and the processor core
 - μ 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 peripheral requests

1.1.4.2 System Control and Clocks (see page 194)

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 memory size, and so on
- Power control
 - On-chip fixed Low Drop-Out (LDO) voltage regulator
 - Hibernation module handles the power-up/down 3.3 V sequencing and control for the core digital logic and analog circuits
 - 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 $16\text{ MHz} \pm 1\%$ frequency at room temperature
 - $16\text{ MHz} \pm 3\%$ across temperature
 - Can be recalibrated with 7-bit trim resolution
 - 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 $30\text{ kHz} \pm 50\%$ frequency, used during power-saving modes
 - 32.768-kHz external oscillator for the Hibernation Module: eliminates need for additional crystal for main clock source
- 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
- MOSC failure

1.1.4.3 Four Programmable Timers (see page 479)

Programmable timers can be used to count or time external events that drive the Timer input pins. 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 General-Purpose Timer Module (GPTM) contains four GPTM blocks with the following functional options:

- Operating modes:
 - 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
 - 16-bit input-edge count- or time-capture modes
 - 16-bit PWM mode with software-programmable output inversion of the PWM signal
- Count up or down
- Eight Capture Compare PWM pins (CCP)
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events
- ADC event trigger
- User-enabled stalling when the microcontroller asserts CPU Halt flag during debug (excluding RTC mode)
- 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 (μ DMA)
 - Dedicated channel for each timer
 - Burst request generated on timer interrupt

1.1.4.4 CCP Pins (see page 486)

Capture Compare PWM pins (CCP) 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.

The LM3S9L97 microcontroller includes eight Capture Compare PWM pins (CCP) that 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.

1.1.4.5 Hibernation Module (see page 301)

The Hibernation module provides logic to switch power off to the main processor and peripherals and to wake on external or time-based events. The Hibernation module includes power-sequencing logic and has the following features:

- 32-bit real-time counter (RTC)
 - Two 32-bit RTC match registers for timed wake-up and interrupt generation
 - RTC predivider trim for making fine adjustments to the clock rate
- Two mechanisms for power control
 - System power control using discrete external regulator
 - On-chip power control using internal switches under register control
- Dedicated pin for waking using an external signal
- RTC operational and hibernation memory valid as long as V_{BAT} is valid
- Low-battery detection, signaling, and interrupt generation
- Clock source from a 32.768-kHz external oscillator or a 4.194304-MHz crystal; 32.768-kHz external oscillator can be used for main controller clock
- 64 32-bit words of non-volatile memory to save state during hibernation
- Programmable interrupts for RTC match, external wake, and low battery events

1.1.4.6 Watchdog Timers (see page 525)

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 Watchdog Timer can generate an interrupt 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 microcontroller 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 LM3S9L97 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 microcontroller asserts the CPU Halt flag during debug

1.1.4.7 Programmable GPIOs (see page 423)

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 0-60 programmable input/output pins. The number of GPIOs available depends on the peripherals being used (see “Signal Tables” on page 1162 for the signals available to each GPIO pin).

- Up to 60 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 5-V-tolerant in input configuration
- Fast toggle capable of a change every two clock cycles
- Two means of port access: either Advanced High-Performance 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

1.1.5 Advanced Motion Control

The LM3S9L97 microcontroller provides motion control functions integrated into the device, including:

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

The following provides more detail on these motion control functions.

1.1.5.1 PWM (see page 1065)

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 LM3S9L97 PWM module consists of three PWM generator blocks and a control block. Each PWM generator block contains one timer (16-bit down or up/down counter), two comparators, a PWM signal generator, a dead-band generator, and an interrupt/ADC-trigger selector. 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:

- Four fault-condition handling input to quickly provide low-latency shutdown and prevent damage to the motor being controlled
- 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
- Synchronization of PWM output enables 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

1.1.5.2 QEI (see page 1137)

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 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, 20 MHz for a 80-MHz system).

The LM3S9L97 microcontroller includes two QEI modules providing control of two motors at the same time with the following features:

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

- Quadrature error detection

1.1.6 Analog

The LM3S9L97 microcontroller provides analog functions integrated into the device, including:

- Two 10-bit Analog-to-Digital Converters (ADC) with 16 analog input channels and sample rate of one million samples/second
- Two analog comparators
- Eight digital comparators
- On-chip voltage regulator

The following provides more detail on these analog functions.

1.1.6.1 ADC (see page 550)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number. The Stellaris ADC module features 10-bit conversion resolution and supports 16 input channels plus an internal temperature sensor. Four buffered sample sequencers allow rapid sampling of up to 16 analog input sources without controller intervention. Each sample sequencer provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequencer priority. Each ADC module has a digital comparator function included that allows the conversion value to be diverted to a comparison unit that provides eight digital comparators.

The LM3S9L97 microcontroller provides two ADC modules with the following features:

- 16 analog input channels
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Maximum sample rate of one million samples/second
- Optional phase shift in sample time programmable from 22.5° to 337.5°
- 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 eight digital comparators
- 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 (μ DMA)
 - Dedicated channel for each sample sequencer
 - ADC module uses burst requests for DMA

1.1.6.2 Analog Comparators (see page 1053)

An analog comparator is a peripheral that compares two analog voltages and provides a logical output that signals the comparison result. The LM3S9L97 microcontroller provides two 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.

The LM3S9L97 microcontroller provides two independent integrated analog comparators with 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 the following voltages:
 - An individual external reference voltage
 - A shared single external reference voltage
 - A shared internal reference voltage

1.1.7 JTAG and ARM Serial Wire Debug (see page 182)

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. Texas Instruments replaces the ARM SW-DP and JTAG-DP with the ARM 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. 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 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

1.1.8 Packaging and Temperature

- Industrial-range 100-pin RoHS-compliant LQFP package
- Industrial-range 108-ball RoHS-compliant BGA package

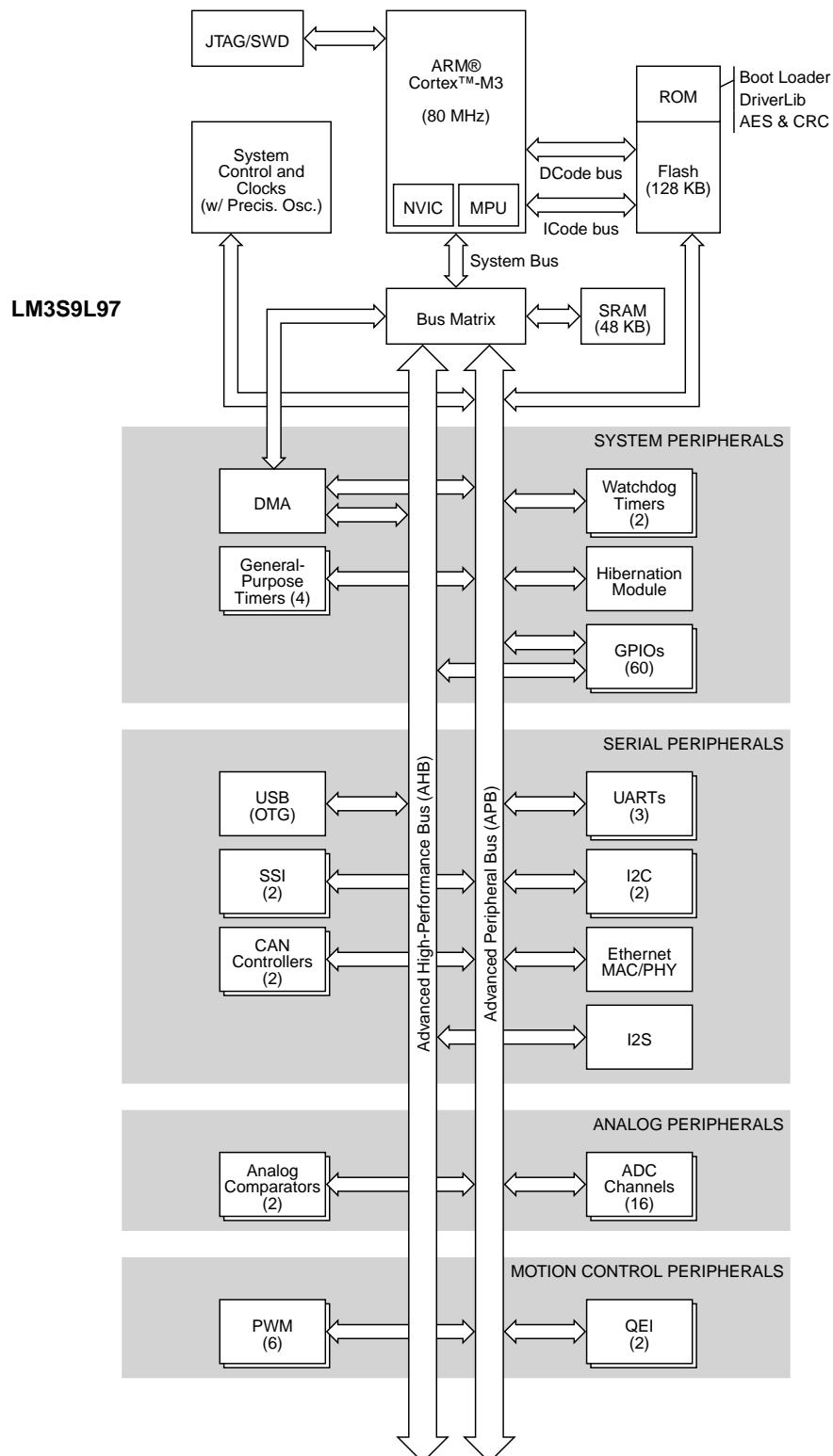
1.2 Target Applications

The Stellaris 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 on page 71 depicts the features on the Stellaris LM3S9L97 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 High-Performance Bus (AHB) bus provides better back-to-back access performance than the APB bus.

Figure 1-1. Stellaris LM3S9L97 Microcontroller High-Level Block Diagram

1.4 Hardware Details

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

- “Pin Diagram” on page 1160
- “Signal Tables” on page 1162
- “Operating Characteristics” on page 1231
- “Electrical Characteristics” on page 1232
- “Package Information” on page 1306

2 The Cortex-M3 Processor

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

- 32-bit ARM® Cortex™-M3 architecture optimized for small-footprint embedded applications
- Outstanding processing performance combined with fast interrupt handling
- Thumb-2 mixed 16-/32-bit instruction set delivers the high performance expected of 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
- Fast code execution permits slower processor clock or increases sleep mode time
- Harvard architecture characterized by separate buses for instruction and data
- Efficient processor core, system and memories
- Hardware division and fast multiplier
- Deterministic, high-performance interrupt handling for time-critical applications
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionality
- Enhanced system debug with extensive breakpoint and trace capabilities
- Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging and tracing
- Migration from the ARM7™ processor family for better performance and power efficiency
- Optimized for single-cycle Flash memory usage
- Ultra-low power consumption with integrated sleep modes
- 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 motor control.

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

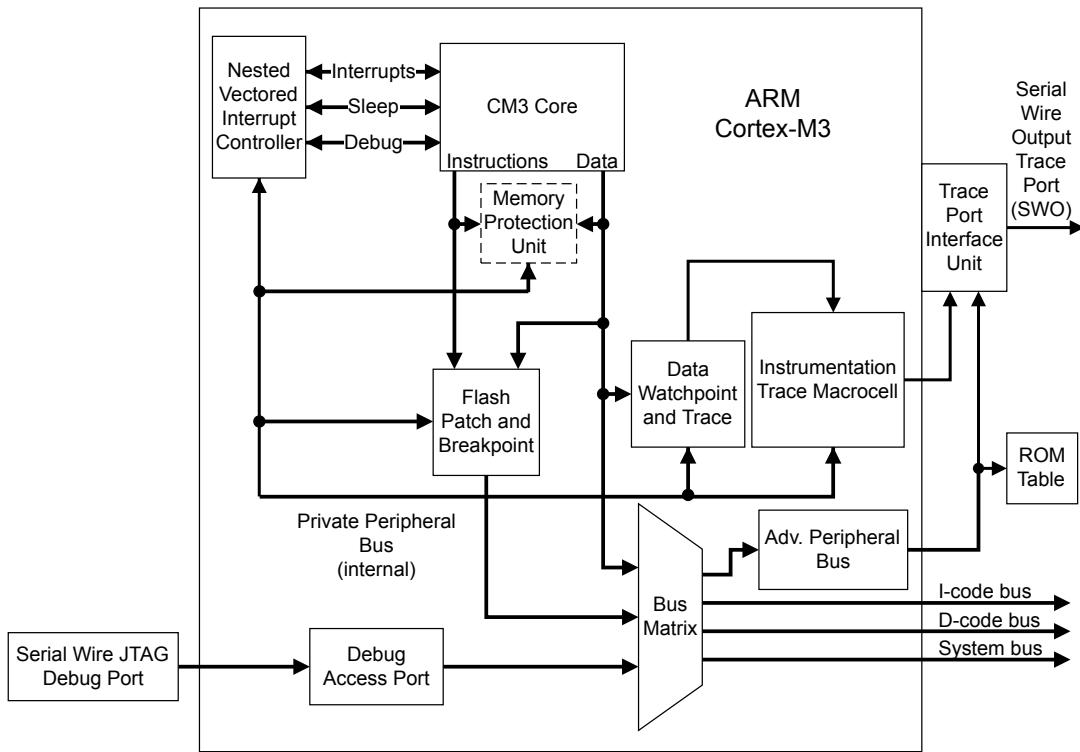
For technical details on the instruction set, see the *Cortex™-M3 Instruction Set Technical User's Manual*.

2.1 Block Diagram

The Cortex-M3 processor is built on a high-performance processor core, with a 3-stage pipeline Harvard architecture, making it ideal for demanding embedded applications. The processor delivers exceptional power efficiency through an efficient instruction set and extensively optimized design, providing high-end processing hardware including single-cycle 32x32 multiplication and dedicated hardware division.

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

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

Figure 2-1. CPU Block Diagram

2.2 Overview

2.2.1 System-Level Interface

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

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

2.2.2 Integrated Configurable Debug

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

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

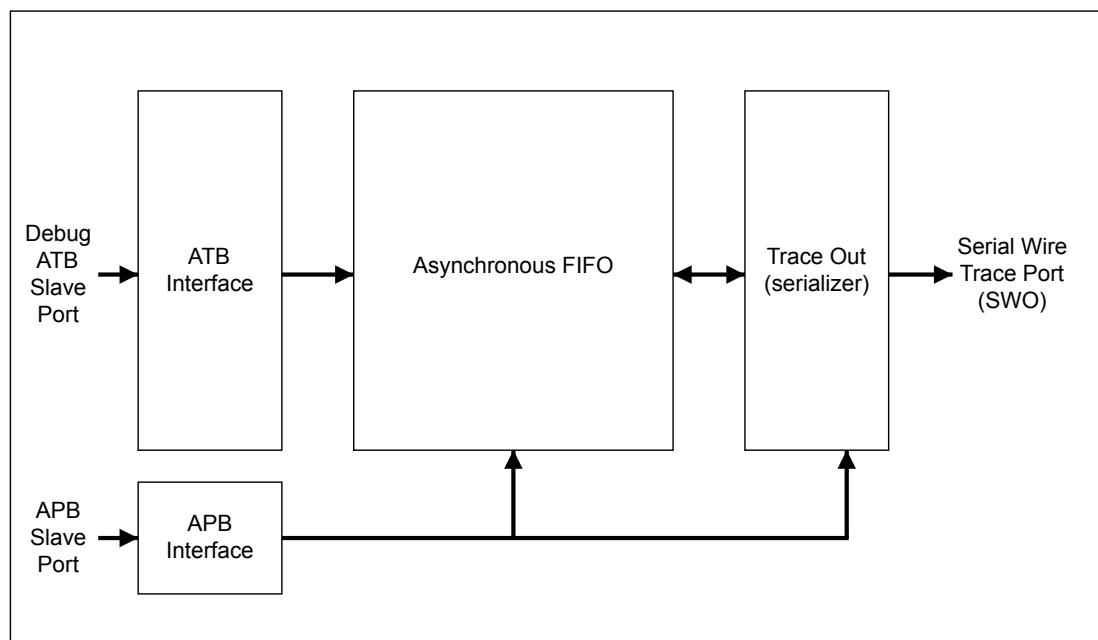
The Flash Patch and Breakpoint Unit (FPB) provides up to eight hardware breakpoint comparators that debuggers can use. The comparators in the FPB also provide remap functions of up to eight words in the program code in the CODE memory region. This enables applications stored in a read-only area of Flash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

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

2.2.3 Trace Port Interface Unit (TPIU)

The TPIU acts as a bridge between the Cortex-M3 trace data from the ITM, and an off-chip Trace Port Analyzer, as shown in Figure 2-2 on page 76.

Figure 2-2. TPIU Block Diagram



2.2.4 Cortex-M3 System Component Details

The Cortex-M3 includes the following system components:

- **SysTick**
A 24-bit count-down timer that can be used as a Real-Time Operating System (RTOS) tick timer or as a simple counter (see “System Timer (SysTick)” on page 116).
- **Nested Vectored Interrupt Controller (NVIC)**
An embedded interrupt controller that supports low latency interrupt processing (see “Nested Vectored Interrupt Controller (NVIC)” on page 117).
- **System Control Block (SCB)**

The programming model interface to the processor. The SCB provides system implementation information and system control, including configuration, control, and reporting of system exceptions (see “System Control Block (SCB)” on page 119).

- Memory Protection Unit (MPU)

Improves system reliability by defining the memory attributes for different memory regions. The MPU provides up to eight different regions and an optional predefined background region (see “Memory Protection Unit (MPU)” on page 119).

2.3 Programming Model

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

2.3.1 Processor Mode and Privilege Levels for Software Execution

The Cortex-M3 has two modes of operation:

- Thread mode

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

- Handler mode

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

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

- Unprivileged

In this mode, software has the following restrictions:

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

- Privileged

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

In Thread mode, the **CONTROL** register (see page 91) controls whether software execution is privileged or unprivileged. In Handler mode, software execution is always privileged.

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

2.3.2 Stacks

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

two stacks: the main stack and the process stack, with independent copies of the stack pointer (see the **SP** register on page 81).

In Thread mode, the **CONTROL** register (see page 91) controls whether the processor uses the main stack or the process stack. In Handler mode, the processor always uses the main stack. The options for processor operations are shown in Table 2-1 on page 78.

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

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

a. See **CONTROL** (page 91).

2.3.3 Register Map

Figure 2-3 on page 78 shows the Cortex-M3 register set. Table 2-2 on page 79 lists the Core registers. The core registers are not memory mapped and are accessed by register name, so the base address is n/a (not applicable) and there is no offset.

Figure 2-3. Cortex-M3 Register Set

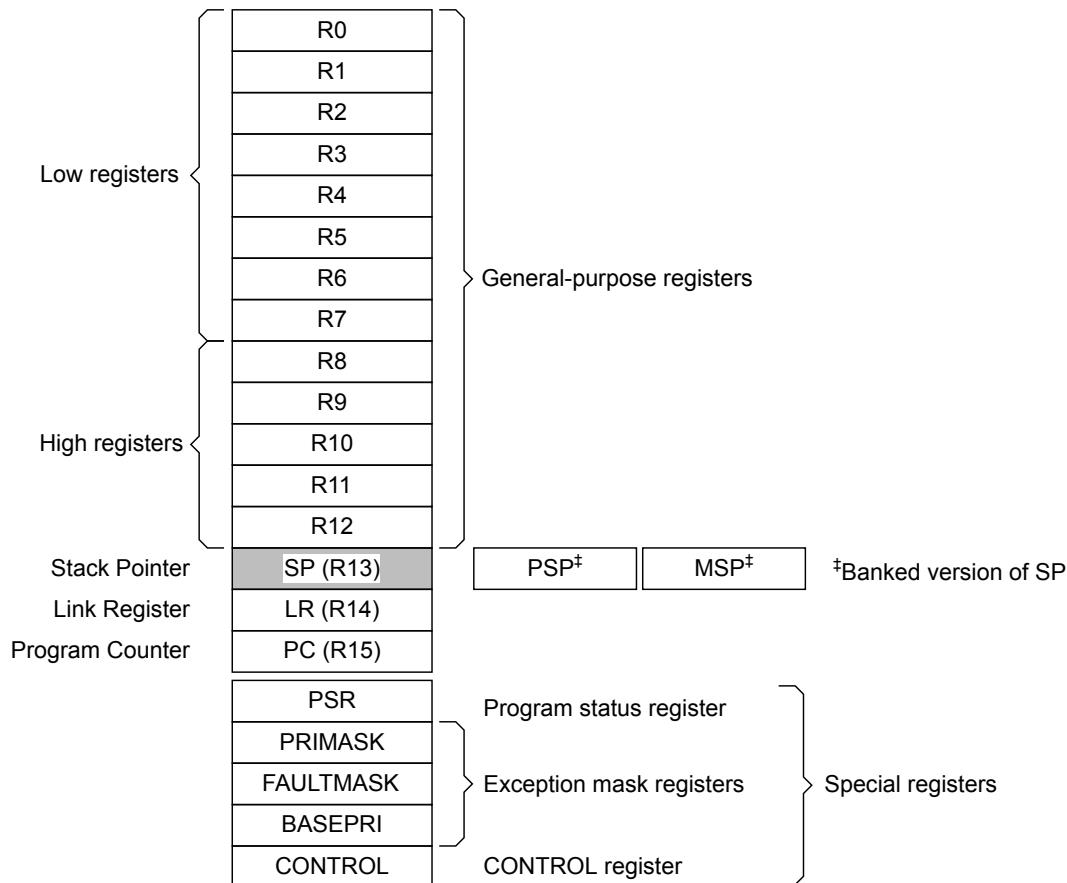


Table 2-2. Processor Register Map

Offset	Name	Type	Reset	Description	See page
-	R0	R/W	-	Cortex General-Purpose Register 0	80
-	R1	R/W	-	Cortex General-Purpose Register 1	80
-	R2	R/W	-	Cortex General-Purpose Register 2	80
-	R3	R/W	-	Cortex General-Purpose Register 3	80
-	R4	R/W	-	Cortex General-Purpose Register 4	80
-	R5	R/W	-	Cortex General-Purpose Register 5	80
-	R6	R/W	-	Cortex General-Purpose Register 6	80
-	R7	R/W	-	Cortex General-Purpose Register 7	80
-	R8	R/W	-	Cortex General-Purpose Register 8	80
-	R9	R/W	-	Cortex General-Purpose Register 9	80
-	R10	R/W	-	Cortex General-Purpose Register 10	80
-	R11	R/W	-	Cortex General-Purpose Register 11	80
-	R12	R/W	-	Cortex General-Purpose Register 12	80
-	SP	R/W	-	Stack Pointer	81
-	LR	R/W	0xFFFF.FFFF	Link Register	82
-	PC	R/W	-	Program Counter	83
-	PSR	R/W	0x0100.0000	Program Status Register	84
-	PRIMASK	R/W	0x0000.0000	Priority Mask Register	88
-	FAULTMASK	R/W	0x0000.0000	Fault Mask Register	89
-	BASEPRI	R/W	0x0000.0000	Base Priority Mask Register	90
-	CONTROL	R/W	0x0000.0000	Control Register	91

2.3.4 Register Descriptions

This section lists and describes the Cortex-M3 registers, in the order shown in Figure 2-3 on page 78. The core registers are not memory mapped and are accessed by register name rather than offset.

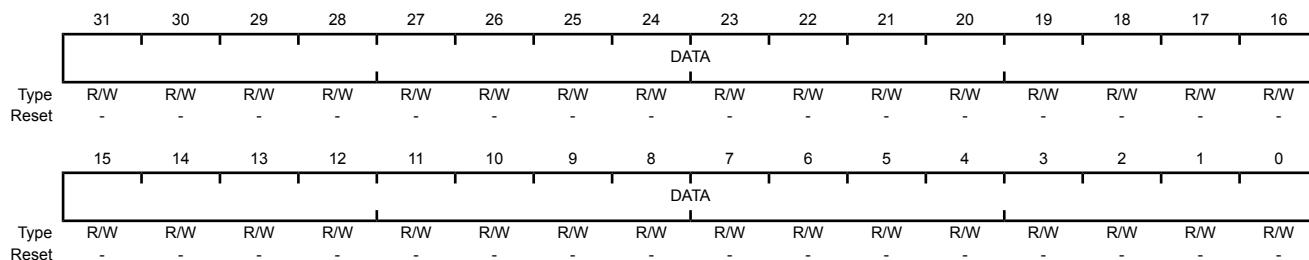
Note: The register type shown in the register descriptions refers to type during program execution in Thread mode and Handler mode. Debug access can differ.

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

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

Cortex General-Purpose Register 0 (R0)

Type R/W, reset -



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

Register 14: Stack Pointer (SP)

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

Stack Pointer (SP)

Type R/W, reset -

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

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

31:0	SP	R/W	-	This field is the address of the stack pointer.
------	----	-----	---	-------------------------------------------------

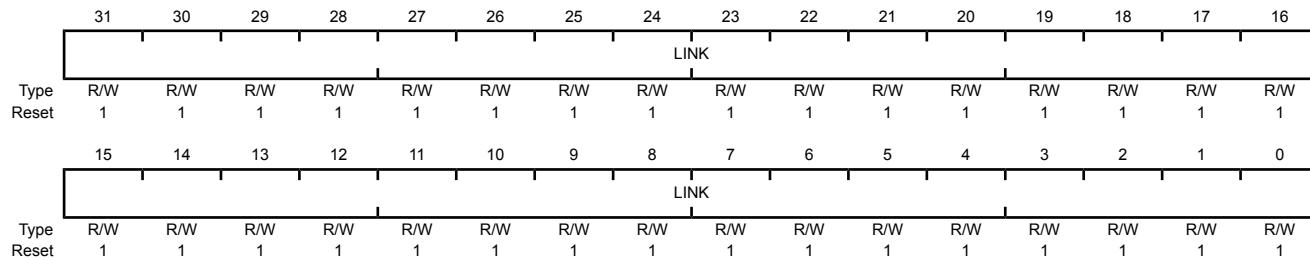
Register 15: Link Register (LR)

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

`EXC_RETURN` is loaded into **LR** on exception entry. See Table 2-10 on page 109 for the values and description.

Link Register (LR)

Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	LINK	R/W	0xFFFF.FFFF	This field is the return address.

Register 16: Program Counter (PC)

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

Program Counter (PC)

Type R/W, reset -

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

Bit/Field Name Type Reset Description

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

Register 17: Program Status Register (PSR)

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

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

- **Application Program Status Register (APSR)**, bits 31:27,
- **Execution Program Status Register (EPSR)**, bits 26:24, 15:10
- **Interrupt Program Status Register (IPSR)**, bits 6:0

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

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

EPSR contains the Thumb state bit and the execution state bits for the If-Then (**IT**) instruction or the Interruptible-Continuable Instruction (**ICI**) field for an interrupted load multiple or store multiple instruction. Attempts to read the **EPSR** directly through application software using the **MSR** instruction always return zero. Attempts to write the **EPSR** using the **MSR** instruction in application software are always ignored. Fault handlers can examine the **EPSR** value in the stacked **PSR** to determine the operation that faulted (see “Exception Entry and Return” on page 107).

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

These registers can be accessed individually or as a combination of any two or all three registers, using the register name as an argument to the **MSR** or **MRS** instructions. For example, all of the registers can be read using **PSR** with the **MRS** instruction, or **APSR** only can be written to using **APSR** with the **MSR** instruction. page 84 shows the possible register combinations for the **PSR**. See the **MRS** and **MSR** instruction descriptions in the *Cortex™-M3 Instruction Set Technical User’s Manual* for more information about how to access the program status registers.

Table 2-3. PSR Register Combinations

Register	Type	Combination
PSR	R/W ^{a, b}	APSR , EPSR , and IPSR
IEPSR	RO	EPSR and IPSR
IAPSR	R/W ^a	APSR and IPSR
EAPSR	R/W ^b	APSR and EPSR

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

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

Program Status Register (PSR)

Type R/W, reset 0x0100.0000

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

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

Bit/Field	Name	Type	Reset	Description
26:25	ICI / IT	RO	0x0	<p>EPSR ICI / IT status</p> <p>These bits, along with bits 15:10, contain the Interruptible-Continuable Instruction (ICI) field for an interrupted load multiple or store multiple instruction or the execution state bits of the IT instruction.</p> <p>When EPSR holds the ICI execution state, bits 26:25 are zero.</p> <p>The If-Then block contains up to four instructions following a 16-bit IT instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See the <i>Cortex™-M3 Instruction Set Technical User's Manual</i> for more information.</p> <p>The value of this field is only meaningful when accessing PSR or EPSR.</p>
24	THUMB	RO	1	<p>EPSR Thumb State</p> <p>This bit indicates the Thumb state and should always be set.</p> <p>The following can clear the THUMB bit:</p> <ul style="list-style-type: none"> ■ The BLX, BX and POP{PC} instructions ■ Restoration from the stacked xPSR value on an exception return ■ Bit 0 of the vector value on an exception entry <p>Attempting to execute instructions when this bit is clear results in a fault or lockup. See “Lockup” on page 111 for more information.</p> <p>The value of this bit is only meaningful when accessing PSR or EPSR.</p>
23:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:10	ICI / IT	RO	0x0	<p>EPSR ICI / IT status</p> <p>These bits, along with bits 26:25, contain the Interruptible-Continuable Instruction (ICI) field for an interrupted load multiple or store multiple instruction or the execution state bits of the IT instruction.</p> <p>When an interrupt occurs during the execution of an LDM, STM, PUSH or POP instruction, the processor stops the load multiple or store multiple instruction operation temporarily and stores the next register operand in the multiple operation to bits 15:12. After servicing the interrupt, the processor returns to the register pointed to by bits 15:12 and resumes execution of the multiple load or store instruction. When EPSR holds the ICI execution state, bits 11:10 are zero.</p> <p>The If-Then block contains up to four instructions following a 16-bit IT instruction. Each instruction in the block is conditional. The conditions for the instructions are either all the same, or some can be the inverse of others. See the <i>Cortex™-M3 Instruction Set Technical User's Manual</i> for more information.</p> <p>The value of this field is only meaningful when accessing PSR or EPSR.</p>
9:7	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.

Bit/Field	Name	Type	Reset	Description
6:0	ISRNUM	RO	0x00	IPSR ISR Number This field contains the exception type number of the current Interrupt Service Routine (ISR).
				Value Description
			0x00	Thread mode
			0x01	Reserved
			0x02	NMI
			0x03	Hard fault
			0x04	Memory management fault
			0x05	Bus fault
			0x06	Usage fault
			0x07-0x0A	Reserved
			0x0B	SVCALL
			0x0C	Reserved for Debug
			0x0D	Reserved
			0x0E	PendSV
			0x0F	SysTick
			0x10	Interrupt Vector 0
			0x11	Interrupt Vector 1
		
			0x46	Interrupt Vector 54
			0x47-0x7F	Reserved

See “Exception Types” on page 102 for more information.
The value of this field is only meaningful when accessing **PSR** or **IPSR**.

Register 18: Priority Mask Register (PRIMASK)

The **PRIMASK** register prevents activation of all exceptions with programmable priority. Reset, non-maskable interrupt (NMI), and hard fault are the only exceptions with fixed priority. Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. The **MSR** and **MRS** instructions are used to access the **PRIMASK** register, and the **CPS** instruction may be used to change the value of the **PRIMASK** register. See the *Cortex™-M3 Instruction Set Technical User's Manual* for more information on these instructions. For more information on exception priority levels, see “Exception Types” on page 102.

Priority Mask Register (PRIMASK)

Type R/W, reset 0x0000.0000

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

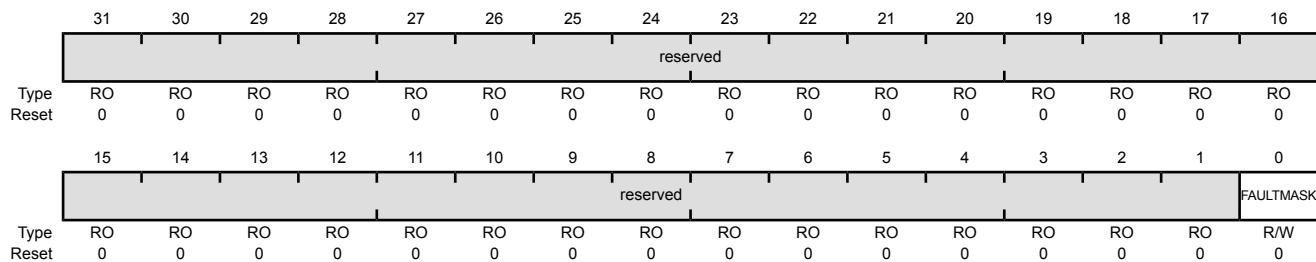
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	PRIMASK	R/W	0	Priority Mask
		Value	Description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

Register 19: Fault Mask Register (FAULTMASK)

The **FAULTMASK** register prevents activation of all exceptions except for the Non-Maskable Interrupt (NMI). Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. The **MSR** and **MRS** instructions are used to access the **FAULTMASK** register, and the **CPS** instruction may be used to change the value of the **FAULTMASK** register. See the *Cortex™-M3 Instruction Set Technical User's Manual* for more information on these instructions. For more information on exception priority levels, see "Exception Types" on page 102.

Fault Mask Register (FAULTMASK)

Type R/W, reset 0x0000.0000



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

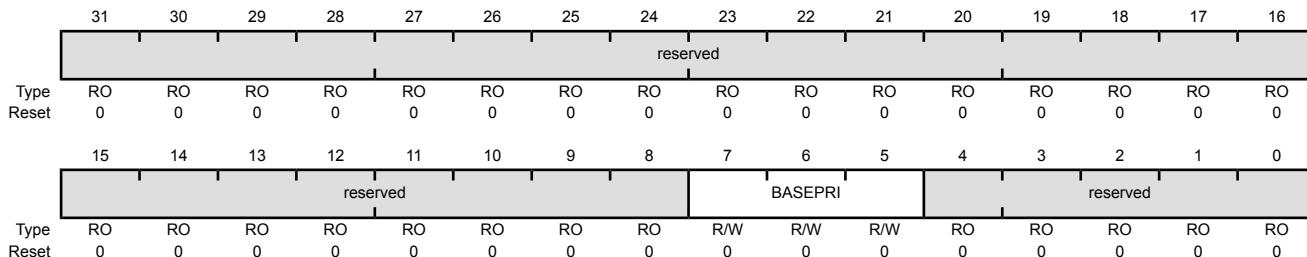
The processor clears the FAULTMASK bit on exit from any exception handler except the NMI handler.

Register 20: Base Priority Mask Register (BASEPRI)

The **BASEPRI** register defines the minimum priority for exception processing. When **BASEPRI** is set to a nonzero value, it prevents the activation of all exceptions with the same or lower priority level as the **BASEPRI** value. Exceptions should be disabled when they might impact the timing of critical tasks. This register is only accessible in privileged mode. For more information on exception priority levels, see “Exception Types” on page 102.

Base Priority Mask Register (BASEPRI)

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	BASEPRI	R/W	0x0	Base Priority Any exception that has a programmable priority level with the same or lower priority as the value of this field is masked. The PRIMASK register can be used to mask all exceptions with programmable priority levels. Higher priority exceptions have lower priority levels.
		Value	Description	
		0x0	All exceptions are unmasked.	
		0x1	All exceptions with priority level 1-7 are masked.	
		0x2	All exceptions with priority level 2-7 are masked.	
		0x3	All exceptions with priority level 3-7 are masked.	
		0x4	All exceptions with priority level 4-7 are masked.	
		0x5	All exceptions with priority level 5-7 are masked.	
		0x6	All exceptions with priority level 6-7 are masked.	
		0x7	All exceptions with priority level 7 are masked.	
4:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 21: Control Register (CONTROL)

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

Handler mode always uses **MSP**, so the processor ignores explicit writes to the **ASP** bit of the **CONTROL** register when in Handler mode. The exception entry and return mechanisms automatically update the **CONTROL** register based on the **EXC_RETURN** value (see Table 2-10 on page 109). In an OS environment, threads running in Thread mode should use the process stack and the kernel and exception handlers should use the main stack. By default, Thread mode uses **MSP**. To switch the stack pointer used in Thread mode to **PSP**, either use the **MSR** instruction to set the **ASP** bit, as detailed in the *Cortex™-M3 Instruction Set Technical User's Manual*, or perform an exception return to Thread mode with the appropriate **EXC_RETURN** value, as shown in Table 2-10 on page 109.

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

Control Register (CONTROL)

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description						
31:2	reserved	RO	0x0000.000	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	ASP	R/W	0	<p>Active Stack Pointer</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>PSP is the current stack pointer.</td> </tr> <tr> <td>0</td> <td>MSP is the current stack pointer</td> </tr> </tbody> </table> <p>In Handler mode, this bit reads as zero and ignores writes. The Cortex-M3 updates this bit automatically on exception return.</p>	Value	Description	1	PSP is the current stack pointer.	0	MSP is the current stack pointer
Value	Description									
1	PSP is the current stack pointer.									
0	MSP is the current stack pointer									
0	TMPL	R/W	0	<p>Thread Mode Privilege Level</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Unprivileged software can be executed in Thread mode.</td> </tr> <tr> <td>0</td> <td>Only privileged software can be executed in Thread mode.</td> </tr> </tbody> </table>	Value	Description	1	Unprivileged software can be executed in Thread mode.	0	Only privileged software can be executed in Thread mode.
Value	Description									
1	Unprivileged software can be executed in Thread mode.									
0	Only privileged software can be executed in Thread mode.									

2.3.5 Exceptions and Interrupts

The Cortex-M3 processor supports interrupts and system exceptions. The processor and the Nested Vectored Interrupt Controller (NVIC) prioritize and handle all exceptions. An exception changes the normal flow of software control. The processor uses Handler mode to handle all exceptions except for reset. See “Exception Entry and Return” on page 107 for more information.

The NVIC registers control interrupt handling. See “Nested Vectored Interrupt Controller (NVIC)” on page 117 for more information.

2.3.6 Data Types

The Cortex-M3 supports 32-bit words, 16-bit halfwords, and 8-bit bytes. The processor also supports 64-bit data transfer instructions. All instruction and data memory accesses are little endian. See “Memory Regions, Types and Attributes” on page 94 for more information.

2.4 Memory Model

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

The memory map for the LM3S9L97 controller is provided in Table 2-4 on page 92. In this manual, register addresses are given as a hexadecimal increment, relative to the module’s base address as shown in the memory map.

The regions for SRAM and peripherals include bit-band regions. Bit-banding provides atomic operations to bit data (see “Bit-Banding” on page 97).

The processor reserves regions of the Private peripheral bus (PPB) address range for core peripheral registers (see “Cortex-M3 Peripherals” on page 116).

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

Table 2-4. Memory Map

Start	End	Description	For details, see page ...
Memory			
0x0000.0000	0x0001.FFFF	On-chip Flash	331
0x0002.0000	0x00FF.FFFF	Reserved	-
0x0100.0000	0x1FFF.FFFF	Reserved for ROM	329
0x2000.0000	0x2000.BFFF	Bit-banded on-chip SRAM	329
0x2000.C000	0x21FF.FFFF	Reserved	-
0x2200.0000	0x2217.FFFF	Bit-band alias of 0x2000.0000 through 0x200F.FFFF	329
0x2218.0000	0x3FFF.FFFF	Reserved	-
FiRM Peripherals			
0x4000.0000	0x4000.0FFF	Watchdog timer 0	528
0x4000.1000	0x4000.1FFF	Watchdog timer 1	528
0x4000.2000	0x4000.3FFF	Reserved	-
0x4000.4000	0x4000.4FFF	GPIO Port A	436
0x4000.5000	0x4000.5FFF	GPIO Port B	436
0x4000.6000	0x4000.6FFF	GPIO Port C	436

Table 2-4. Memory Map (continued)

Start	End	Description	For details, see page ...
0x4000.7000	0x4000.7FFF	GPIO Port D	436
0x4000.8000	0x4000.8FFF	SSI0	703
0x4000.9000	0x4000.9FFF	SSI1	703
0x4000.A000	0x4000.BFFF	Reserved	-
0x4000.C000	0x4000.CFFF	UART0	640
0x4000.D000	0x4000.DFFF	UART1	640
0x4000.E000	0x4000.EFFF	UART2	640
0x4000.F000	0x4001.FFFF	Reserved	-
Peripherals			
0x4002.0000	0x4002.0FFF	I ² C 0	746
0x4002.1000	0x4002.1FFF	I ² C 1	746
0x4002.2000	0x4002.3FFF	Reserved	-
0x4002.4000	0x4002.4FFF	GPIO Port E	436
0x4002.5000	0x4002.5FFF	GPIO Port F	436
0x4002.6000	0x4002.6FFF	GPIO Port G	436
0x4002.7000	0x4002.7FFF	GPIO Port H	436
0x4002.8000	0x4002.8FFF	PWM	1078
0x4002.9000	0x4002.BFFF	Reserved	-
0x4002.C000	0x4002.CFFF	QEIO	1143
0x4002.D000	0x4002.DFFF	QEII	1143
0x4002.E000	0x4002.FFFF	Reserved	-
0x4003.0000	0x4003.0FFF	Timer 0	494
0x4003.1000	0x4003.1FFF	Timer 1	494
0x4003.2000	0x4003.2FFF	Timer 2	494
0x4003.3000	0x4003.3FFF	Timer 3	494
0x4003.4000	0x4003.7FFF	Reserved	-
0x4003.8000	0x4003.8FFF	ADC0	570
0x4003.9000	0x4003.9FFF	ADC1	570
0x4003.A000	0x4003.BFFF	Reserved	-
0x4003.C000	0x4003.CFFF	Analog Comparators	1053
0x4003.D000	0x4003.DFFF	GPIO Port J	436
0x4003.E000	0x4003.FFFF	Reserved	-
0x4004.0000	0x4004.0FFF	CAN0 Controller	823
0x4004.1000	0x4004.1FFF	CAN1 Controller	823
0x4004.2000	0x4004.7FFF	Reserved	-
0x4004.8000	0x4004.8FFF	Ethernet Controller	867
0x4004.9000	0x4004.FFFF	Reserved	-
0x4005.0000	0x4005.0FFF	USB	941
0x4005.1000	0x4005.3FFF	Reserved	-
0x4005.4000	0x4005.4FFF	I ² S0	779
0x4005.5000	0x4005.7FFF	Reserved	-

Table 2-4. Memory Map (continued)

Start	End	Description	For details, see page ...
0x4005.8000	0x4005.8FFF	GPIO Port A (AHB aperture)	436
0x4005.9000	0x4005.9FFF	GPIO Port B (AHB aperture)	436
0x4005.A000	0x4005.AFFF	GPIO Port C (AHB aperture)	436
0x4005.B000	0x4005.BFFF	GPIO Port D (AHB aperture)	436
0x4005.C000	0x4005.CFFF	GPIO Port E (AHB aperture)	436
0x4005.D000	0x4005.DFFF	GPIO Port F (AHB aperture)	436
0x4005.E000	0x4005.EFFF	GPIO Port G (AHB aperture)	436
0x4005.F000	0x4005.FFFF	GPIO Port H (AHB aperture)	436
0x4006.0000	0x4006.0FFF	GPIO Port J (AHB aperture)	436
0x4006.1000	0x400F.BFFF	Reserved	-
0x400F.C000	0x400F.CFFF	Hibernation Module	311
0x400F.D000	0x400F.DFFF	Flash memory control	337
0x400F.E000	0x400F.EFFF	System control	214
0x400F.F000	0x400F.FFFF	μ DMA	386
0x4010.0000	0x41FF.FFFF	Reserved	-
0x4200.0000	0x43FF.FFFF	Bit-banded alias of 0x4000.0000 through 0x400F.FFFF	-
0x4400.0000	0xDFFF.FFFF	Reserved	-
Private Peripheral Bus			
0xE000.0000	0xE000.0FFF	Instrumentation Trace Macrocell (ITM)	75
0xE000.1000	0xE000.1FFF	Data Watchpoint and Trace (DWT)	75
0xE000.2000	0xE000.2FFF	Flash Patch and Breakpoint (FPB)	75
0xE000.3000	0xE000.DFFF	Reserved	-
0xE000.E000	0xE000.EFFF	Cortex-M3 Peripherals (SysTick, NVIC, SCB, and MPU)	101
0xE000.F000	0xE003.FFFF	Reserved	-
0xE004.0000	0xE004.0FFF	Trace Port Interface Unit (TPIU)	76
0xE004.1000	0xFFFF.FFFF	Reserved	-

2.4.1 Memory Regions, Types and Attributes

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

The memory types are:

- Normal: The processor can re-order transactions for efficiency and perform speculative reads.
- Device: The processor preserves transaction order relative to other transactions to Device or Strongly Ordered memory.
- Strongly Ordered: The processor preserves transaction order relative to all other transactions.

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

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

2.4.2 Memory System Ordering of Memory Accesses

For most memory accesses caused by explicit memory access instructions, the memory system does not guarantee that the order in which the accesses complete matches the program order of the instructions, providing the order does not affect the behavior of the instruction sequence. Normally, if correct program execution depends on two memory accesses completing in program order, software must insert a memory barrier instruction between the memory access instructions (see “Software Ordering of Memory Accesses” on page 96).

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

2.4.3 Behavior of Memory Accesses

Table 2-5 on page 95 shows the behavior of accesses to each region in the memory map. See “Memory Regions, Types and Attributes” on page 94 for more information on memory types and the XN attribute. Stellaris devices may have reserved memory areas within the address ranges shown below (refer to Table 2-4 on page 92 for more information).

Table 2-5. Memory Access Behavior

Address Range	Memory Region	Memory Type	Execute Never (XN)	Description
0x0000.0000 - 0x1FFF.FFFF	Code	Normal	-	This executable region is for program code. Data can also be stored here.
0x2000.0000 - 0x3FFF.FFFF	SRAM	Normal	-	This executable region is for data. Code can also be stored here. This region includes bit band and bit band alias areas (see Table 2-6 on page 97).
0x4000.0000 - 0x5FFF.FFFF	Peripheral	Device	XN	This region includes bit band and bit band alias areas (see Table 2-7 on page 97).
0x6000.0000 - 0x9FFF.FFFF	External RAM	Normal	-	This executable region is for data.
0xA000.0000 - 0xDFFF.FFFF	External device	Device	XN	This region is for external device memory.
0xE000.0000- 0xE00F.FFFF	Private peripheral bus	Strongly Ordered	XN	This region includes the NVIC, system timer, and system control block.
0xE010.0000- 0xFFFF.FFFF	Reserved	-	-	-

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

The MPU can override the default memory access behavior described in this section. For more information, see “Memory Protection Unit (MPU)” on page 119.

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

2.4.4 Software Ordering of Memory Accesses

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

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

“Memory System Ordering of Memory Accesses” on page 95 describes the cases where the memory system guarantees the order of memory accesses. Otherwise, if the order of memory accesses is critical, software must include memory barrier instructions to force that ordering. The Cortex-M3 has the following memory barrier instructions:

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

Memory barrier instructions can be used in the following situations:

- MPU programming
 - If the MPU settings are changed and the change must be effective on the very next instruction, use a `DSB` instruction to ensure the effect of the MPU takes place immediately at the end of context switching.
 - Use an `ISB` instruction to ensure the new MPU setting takes effect immediately after programming the MPU region or regions, if the MPU configuration code was accessed using a branch or call. If the MPU configuration code is entered using exception mechanisms, then an `ISB` instruction is not required.

- Vector table

If the program changes an entry in the vector table and then enables the corresponding exception, use a `DMB` instruction between the operations. The `DMB` instruction ensures that if the exception is taken immediately after being enabled, the processor uses the new exception vector.

- Self-modifying code

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

- Memory map switching

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

- Dynamic exception priority change

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

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

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

2.4.5 Bit-Banding

A bit-band region maps each word in a bit-band alias region to a single bit in the bit-band region. The bit-band regions occupy the lowest 1 MB of the SRAM and peripheral memory regions. Accesses to the 32-MB SRAM alias region map to the 1-MB SRAM bit-band region, as shown in Table 2-6 on page 97. Accesses to the 32-MB peripheral alias region map to the 1-MB peripheral bit-band region, as shown in Table 2-7 on page 97. For the specific address range of the bit-band regions, see Table 2-4 on page 92.

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

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

Table 2-6. SRAM Memory Bit-Banding Regions

Address Range	Memory Region	Instruction and Data Accesses
0x2000.0000 - 0x200F.FFFF	SRAM bit-band region	Direct accesses to this memory range behave as SRAM memory accesses, but this region is also bit addressable through bit-band alias.
0x2200.0000 - 0x23FF.FFFF	SRAM bit-band alias	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not remapped.

Table 2-7. Peripheral Memory Bit-Banding Regions

Address Range	Memory Region	Instruction and Data Accesses
0x4000.0000 - 0x400F.FFFF	Peripheral bit-band region	Direct accesses to this memory range behave as peripheral memory accesses, but this region is also bit addressable through bit-band alias.
0x4200.0000 - 0x43FF.FFFF	Peripheral bit-band alias	Data accesses to this region are remapped to bit band region. A write operation is performed as read-modify-write. Instruction accesses are not permitted.

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

```
bit_word_offset = (byte_offset x 32) + (bit_number x 4)  
bit_word_addr = bit_band_base + bit_word_offset
```

where:

bit_word_offset

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

bit_word_addr

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

bit_band_base

The starting address of the alias region.

byte_offset

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

bit_number

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

Figure 2-4 on page 99 shows examples of bit-band mapping between the SRAM bit-band alias region and the SRAM bit-band region:

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

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

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

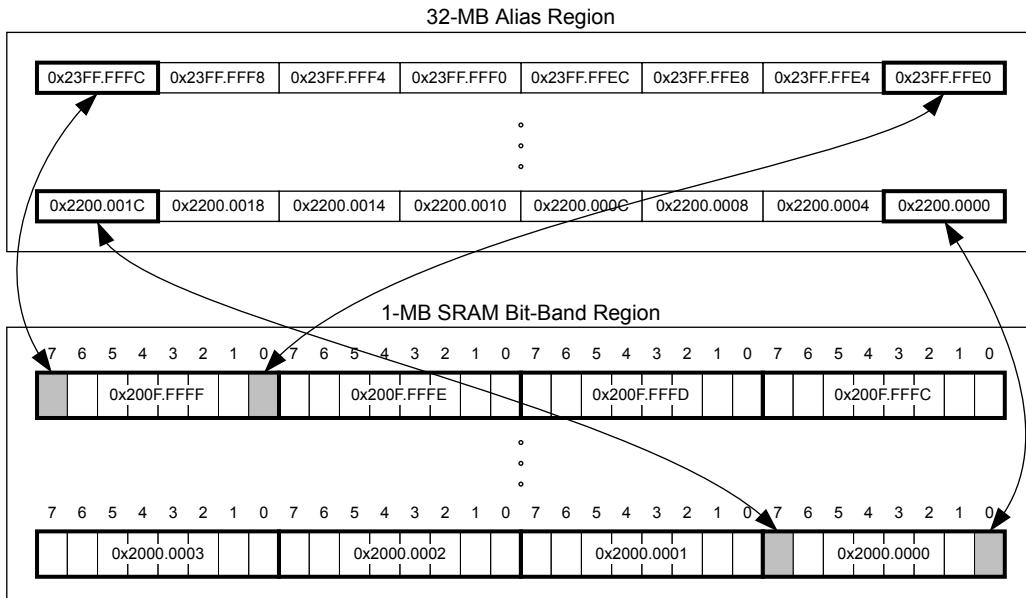
$$0x23FF.FFFC = 0x2200.0000 + (0x000F.FFFF * 32) + (7 * 4)$$

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

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

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

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

Figure 2-4. Bit-Band Mapping

2.4.5.1 Directly Accessing an Alias Region

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

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

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

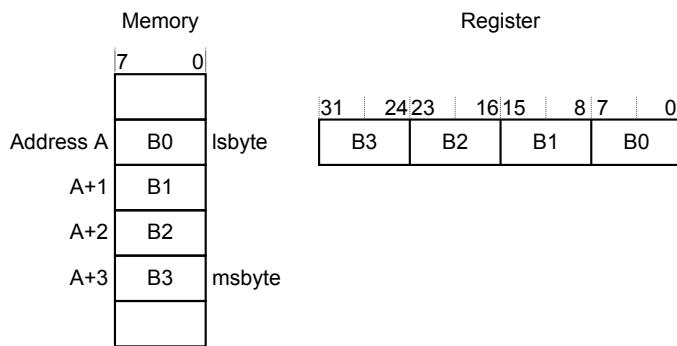
When reading a word in the alias region, 0x0000.0000 indicates that the targeted bit in the bit-band region is clear and 0x0000.0001 indicates that the targeted bit in the bit-band region is set.

2.4.5.2 Directly Accessing a Bit-Band Region

“Behavior of Memory Accesses” on page 95 describes the behavior of direct byte, halfword, or word accesses to the bit-band regions.

2.4.6 Data Storage

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word. Data is stored in little-endian format, with the least-significant byte (lsbyte) of a word stored at the lowest-numbered byte, and the most-significant byte (msbyte) stored at the highest-numbered byte. Figure 2-5 on page 100 illustrates how data is stored.

Figure 2-5. Data Storage

2.4.7 Synchronization Primitives

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

A pair of synchronization primitives consists of:

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

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

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

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

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

1. Use a Load-Exclusive instruction to read the value of the location.
2. Update the value, as required.
3. Use a Store-Exclusive instruction to attempt to write the new value back to the memory location, and test the returned status bit. If the status bit is clear, the read-modify-write completed successfully; if the status bit is set, no write was performed, which indicates that the value returned at step 1 might be out of date. The software must retry the read-modify-write sequence.

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

1. Use a Load-Exclusive instruction to read from the semaphore address to check whether the semaphore is free.

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

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

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

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

2.5 Exception Model

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

Table 2-8 on page 103 lists all exception types. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 51 interrupts (listed in Table 2-9 on page 104).

Priorities on the system handlers are set with the NVIC **System Handler Priority n (SYSPRIn)** registers. Interrupts are enabled through the NVIC **Interrupt Set Enable n (ENn)** register and prioritized with the NVIC **Interrupt Priority n (PRIn)** registers. Priorities can be grouped by splitting priority levels into preemption priorities and subpriorities. All the interrupt registers are described in “Nested Vectored Interrupt Controller (NVIC)” on page 117.

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

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

See “Nested Vectored Interrupt Controller (NVIC)” on page 117 for more information on exceptions and interrupts.

2.5.1 Exception States

Each exception is in one of the following states:

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

2.5.2 Exception Types

The exception types are:

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

An unaligned address on a word or halfword memory access or division by zero can cause a usage fault when the core is properly configured.

- **SVCALL.** A supervisor call (SVC) is an exception that is triggered by the SVC instruction. In an OS environment, applications can use SVC instructions to access OS kernel functions and device drivers.
- **Debug Monitor.** This exception is caused by the debug monitor (when not halting). This exception is only active when enabled. This exception does not activate if it is a lower priority than the current activation.
- **PendSV.** PendSV is a pendable, interrupt-driven request for system-level service. In an OS environment, use PendSV for context switching when no other exception is active. PendSV is triggered using the **Interrupt Control and State (INTCTRL)** register.
- **SysTick.** A SysTick exception is an exception that the system timer generates when it reaches zero when it is enabled to generate an interrupt. Software can also generate a SysTick exception using the **Interrupt Control and State (INTCTRL)** register. In an OS environment, the processor can use this exception as system tick.
- **Interrupt (IRQ).** An interrupt, or IRQ, is an exception signaled by a peripheral or generated by a software request and fed through the NVIC (prioritized). All interrupts are asynchronous to instruction execution. In the system, peripherals use interrupts to communicate with the processor. Table 2-9 on page 104 lists the interrupts on the LM3S9L97 controller.

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

Privileged software can disable the exceptions that Table 2-8 on page 103 shows as having configurable priority (see the **SYSHNDCTRL** register on page 160 and the **DIS0** register on page 133).

For more information about hard faults, memory management faults, bus faults, and usage faults, see “Fault Handling” on page 109.

Table 2-8. Exception Types

Exception Type	Vector Number	Priority ^a	Vector Address or Offset ^b	Activation
-	0	-	0x0000.0000	Stack top is loaded from the first entry of the vector table on reset.
Reset	1	-3 (highest)	0x0000.0004	Asynchronous
Non-Maskable Interrupt (NMI)	2	-2	0x0000.0008	Asynchronous
Hard Fault	3	-1	0x0000.000C	-
Memory Management	4	programmable ^c	0x0000.0010	Synchronous
Bus Fault	5	programmable ^c	0x0000.0014	Synchronous when precise and asynchronous when imprecise
Usage Fault	6	programmable ^c	0x0000.0018	Synchronous
-	7-10	-	-	Reserved
SVCALL	11	programmable ^c	0x0000.002C	Synchronous
Debug Monitor	12	programmable ^c	0x0000.0030	Synchronous
-	13	-	-	Reserved
PendSV	14	programmable ^c	0x0000.0038	Asynchronous

Table 2-8. Exception Types (continued)

Exception Type	Vector Number	Priority ^a	Vector Address or Offset ^b	Activation
SysTick	15	programmable ^c	0x0000.003C	Asynchronous
Interrupts	16 and above	programmable ^d	0x0000.0040 and above	Asynchronous

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

b. See “Vector Table” on page 105.

c. See **SYSPRI1** on page 157.

d. See **PRIn** registers on page 141.

Table 2-9. Interrupts

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
0-15	-	0x0000.0000 - 0x0000.003C	Processor exceptions
16	0	0x0000.0040	GPIO Port A
17	1	0x0000.0044	GPIO Port B
18	2	0x0000.0048	GPIO Port C
19	3	0x0000.004C	GPIO Port D
20	4	0x0000.0050	GPIO Port E
21	5	0x0000.0054	UART0
22	6	0x0000.0058	UART1
23	7	0x0000.005C	SSI0
24	8	0x0000.0060	I ² C0
25	9	0x0000.0064	PWM Fault
26	10	0x0000.0068	PWM Generator 0
27	11	0x0000.006C	PWM Generator 1
28	12	0x0000.0070	PWM Generator 2
29	13	0x0000.0074	QEIO
30	14	0x0000.0078	ADC0 Sequence 0
31	15	0x0000.007C	ADC0 Sequence 1
32	16	0x0000.0080	ADC0 Sequence 2
33	17	0x0000.0084	ADC0 Sequence 3
34	18	0x0000.0088	Watchdog Timers 0 and 1
35	19	0x0000.008C	Timer 0A
36	20	0x0000.0090	Timer 0B
37	21	0x0000.0094	Timer 1A
38	22	0x0000.0098	Timer 1B
39	23	0x0000.009C	Timer 2A
40	24	0x0000.00A0	Timer 2B
41	25	0x0000.00A4	Analog Comparator 0
42	26	0x0000.00A8	Analog Comparator 1
43	27	Reserved	
44	28	0x0000.00B0	System Control
45	29	0x0000.00B4	Flash Memory Control

Table 2-9. Interrupts (continued)

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Vector Address or Offset	Description
46	30	0x0000.00B8	GPIO Port F
47	31	0x0000.00BC	GPIO Port G
48	32	0x0000.00C0	GPIO Port H
49	33	0x0000.00C4	UART2
50	34	0x0000.00C8	SSI1
51	35	0x0000.00CC	Timer 3A
52	36	0x0000.00D0	Timer 3B
53	37	0x0000.00D4	I ² C1
54	38	0x0000.00D8	QEI1
55	39	0x0000.00DC	CAN0
56	40	0x0000.00E0	CAN1
57	41	Reserved	
58	42	0x0000.00E8	Ethernet Controller
59	43	0x0000.00EC	Hibernation Module
60	44	0x0000.00F0	USB
61	45	Reserved	
62	46	0x0000.00F8	μDMA Software
63	47	0x0000.00FC	μDMA Error
64	48	0x0000.0100	ADC1 Sequence 0
65	49	0x0000.0104	ADC1 Sequence 1
66	50	0x0000.0108	ADC1 Sequence 2
67	51	0x0000.010C	ADC1 Sequence 3
68	52	0x0000.0110	I ² S0
69	53	Reserved	
70	54	0x0000.0118	GPIO Port J

2.5.3 Exception Handlers

The processor handles exceptions using:

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

2.5.4 Vector Table

The vector table contains the reset value of the stack pointer and the start addresses, also called exception vectors, for all exception handlers. The vector table is constructed using the vector address or offset shown in Table 2-8 on page 103. Figure 2-6 on page 106 shows the order of the exception vectors in the vector table. The least-significant bit of each vector must be 1, indicating that the exception handler is Thumb code

Figure 2-6. Vector table

Exception number	IRQ number	Offset	Vector
70	54	0x0118	IRQ54
.	.	.	.
.	.	0x004C	
18	2	0x0048	IRQ2
17	1	0x0044	IRQ1
16	0	0x0040	IRQ0
15	-1	0x003C	Systick
14	-2	0x0038	PendSV
13			Reserved
12			Reserved for Debug
11	-5	0x002C	SVCall
10			
9			Reserved
8			
7			
6	-10	0x0018	Usage fault
5	-11	0x0014	Bus fault
4	-12	0x0010	Memory management fault
3	-13	0x000C	Hard fault
2	-14	0x0008	NMI
1		0x0004	Reset
		0x0000	Initial SP value

On system reset, the vector table is fixed at address 0x0000.0000. Privileged software can write to the **Vector Table Offset (VTABLE)** register to relocate the vector table start address to a different memory location, in the range 0x0000.0200 to 0x3FFF.FE00 (see “Vector Table” on page 105). Note that when configuring the **VTABLE** register, the offset must be aligned on a 512-byte boundary.

2.5.5 Exception Priorities

As Table 2-8 on page 103 shows, all exceptions have an associated priority, with a lower priority value indicating a higher priority and configurable priorities for all exceptions except Reset, Hard fault, and NMI. If software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities, see page 157 and page 141.

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

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

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

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

2.5.6 Interrupt Priority Grouping

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

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

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

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

For information about splitting the interrupt priority fields into group priority and subpriority, see page 151.

2.5.7 Exception Entry and Return

Descriptions of exception handling use the following terms:

- **Preemption.** When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled. See “Interrupt Priority Grouping” on page 107 for more information about preemption by an interrupt. When one exception preempts another, the exceptions are called nested exceptions. See “Exception Entry” on page 108 for more information.
- **Return.** Return occurs when the exception handler is completed, and there is no pending exception with sufficient priority to be serviced and the completed exception handler was not handling a late-arriving exception. The processor pops the stack and restores the processor state to the state it had before the interrupt occurred. See “Exception Return” on page 108 for more information.
- **Tail-Chaining.** This mechanism speeds up exception servicing. On completion of an exception handler, if there is a pending exception that meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler.
- **Late-Arriving.** This mechanism speeds up preemption. If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher priority exception and initiates the vector fetch for that exception. State saving is not affected by late arrival because the state saved is the same for both exceptions. Therefore, the state saving continues uninterrupted. The processor can accept a late arriving exception until the first instruction of the exception handler of the original exception enters the execute stage of the processor. On

return from the exception handler of the late-arriving exception, the normal tail-chaining rules apply.

2.5.7.1 Exception Entry

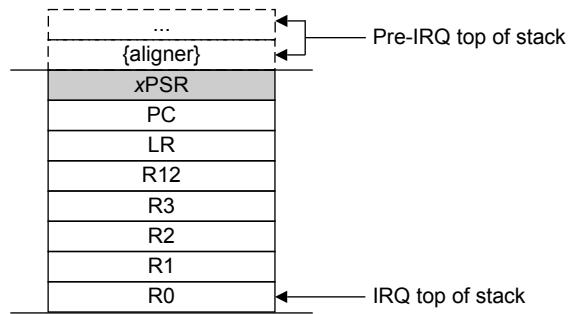
Exception entry occurs when there is a pending exception with sufficient priority and either the processor is in Thread mode or the new exception is of higher priority than the exception being handled, in which case the new exception preempts the original exception.

When one exception preempts another, the exceptions are nested.

Sufficient priority means the exception has more priority than any limits set by the mask registers (see **PRIMASK** on page 88, **FAULTMASK** on page 89, and **BASEPRI** on page 90). An exception with less priority than this is pending but is not handled by the processor.

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

Figure 2-7. Exception Stack Frame



Immediately after stacking, the stack pointer indicates the lowest address in the stack frame.

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

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

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

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

2.5.7.2 Exception Return

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

- An **LDM** or **POP** instruction that loads the **PC**

- A **BX** instruction using any register
- An **LDR** instruction with the **PC** as the destination

EXC_RETURN is the value loaded into the **LR** on exception entry. The exception mechanism relies on this value to detect when the processor has completed an exception handler. The lowest four bits of this value provide information on the return stack and processor mode. Table 2-10 on page 109 shows the **EXC_RETURN** values with a description of the exception return behavior.

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

Table 2-10. Exception Return Behavior

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

2.6 Fault Handling

Faults are a subset of the exceptions (see “Exception Model” on page 101). The following conditions generate a fault:

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

2.6.1 Fault Types

Table 2-11 on page 109 shows the types of fault, the handler used for the fault, the corresponding fault status register, and the register bit that indicates the fault has occurred. See page 164 for more information about the fault status registers.

Table 2-11. Faults

Fault	Handler	Fault Status Register	Bit Name
Bus error on a vector read	Hard fault	Hard Fault Status (HFAULTSTAT)	VECT
Fault escalated to a hard fault	Hard fault	Hard Fault Status (HFAULTSTAT)	FORCED

Table 2-11. Faults (continued)

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

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

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

2.6.2 Fault Escalation and Hard Faults

All fault exceptions except for hard fault have configurable exception priority (see **SYSPRI1** on page 157). Software can disable execution of the handlers for these faults (see **SYSHNDCTRL** on page 160).

Usually, the exception priority, together with the values of the exception mask registers, determines whether the processor enters the fault handler, and whether a fault handler can preempt another fault handler as described in “Exception Model” on page 101.

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

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

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

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

2.6.3 Fault Status Registers and Fault Address Registers

The fault status registers indicate the cause of a fault. For bus faults and memory management faults, the fault address register indicates the address accessed by the operation that caused the fault, as shown in Table 2-12 on page 111.

Table 2-12. Fault Status and Fault Address Registers

Handler	Status Register Name	Address Register Name	Register Description
Hard fault	Hard Fault Status (HFAULTSTAT)	-	page 170
Memory management fault	Memory Management Fault Status (MFAULTSTAT)	Memory Management Fault Address (MMADDR)	page 164 page 171
Bus fault	Bus Fault Status (BFAULTSTAT)	Bus Fault Address (FAULTADDR)	page 164 page 172
Usage fault	Usage Fault Status (UFAULTSTAT)	-	page 164

2.6.4 Lockup

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

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

2.7 Power Management

The Cortex-M3 processor sleep modes reduce power consumption:

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

The **SLEEPDEEP** bit of the **System Control (SYSCTRL)** register selects which sleep mode is used (see page 153). For more information about the behavior of the sleep modes, see “System Control” on page 211.

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

2.7.1 Entering Sleep Modes

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

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

2.7.1.1 Wait for Interrupt

The wait for interrupt instruction, **WFI**, causes immediate entry to sleep mode unless the wake-up condition is true (see “Wake Up from WFI or Sleep-on-Exit” on page 112). When the processor executes a **WFI** instruction, it stops executing instructions and enters sleep mode. See the *Cortex™-M3 Instruction Set Technical User's Manual* for more information.

2.7.1.2 Wait for Event

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

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

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

2.7.1.3 Sleep-on-Exit

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

2.7.2 Wake Up from Sleep Mode

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

2.7.2.1 Wake Up from WFI or Sleep-on-Exit

Normally, the processor wakes up only when it detects an exception with sufficient priority to cause exception entry. Some embedded systems might have to execute system restore tasks after the processor wakes up and before executing an interrupt handler. Entry to the interrupt handler can be delayed by setting the **PRIMASK** bit and clearing the **FAULTMASK** bit. If an interrupt arrives that is enabled and has a higher priority than current exception priority, the processor wakes up but does not execute the interrupt handler until the processor clears **PRIMASK**. For more information about **PRIMASK** and **FAULTMASK**, see page 88 and page 89.

2.7.2.2 Wake Up from WFE

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

In addition, if the **SEVONPEND** bit in the **SYSCtrl** register is set, any new pending interrupt triggers an event and wakes up the processor, even if the interrupt is disabled or has insufficient priority to cause exception entry. For more information about **SYSCtrl**, see page 153.

2.8 Instruction Set Summary

The processor implements a version of the Thumb instruction set. Table 2-13 on page 113 lists the supported instructions.

Note: In Table 2-13 on page 113:

- Angle brackets, **<>**, enclose alternative forms of the operand

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

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

Table 2-13. Cortex-M3 Instruction Summary

Mnemonic	Operands	Brief Description	Flags
ADC, ADCS	{Rd, } Rn , Op2	Add with carry	N,Z,C,V
ADD, ADDS	{Rd, } Rn , Op2	Add	N,Z,C,V
ADD, ADDW	{Rd, } Rn , #imm12	Add	N,Z,C,V
ADR	Rd , label	Load PC-relative address	-
AND, ANDS	{Rd, } Rn , Op2	Logical AND	N,Z,C
ASR, ASRS	Rd , Rm , <Rs #n>	Arithmetic shift right	N,Z,C
B	label	Branch	-
BFC	Rd , #lsb , #width	Bit field clear	-
BFI	Rd , Rn , #lsb , #width	Bit field insert	-
BIC, BICS	{Rd, } Rn , Op2	Bit clear	N,Z,C
BKPT	#imm	Breakpoint	-
BL	label	Branch with link	-
BLX	Rm	Branch indirect with link	-
BX	Rm	Branch indirect	-
CBNZ	Rn , label	Compare and branch if non-zero	-
CBZ	Rn , label	Compare and branch if zero	-
CLREX	-	Clear exclusive	-
CLZ	Rd , Rm	Count leading zeros	-
CMN	Rn , Op2	Compare negative	N,Z,C,V
CMP	Rn , Op2	Compare	N,Z,C,V
CPSID	iflags	Change processor state, disable interrupts	-
CPSIE	iflags	Change processor state, enable interrupts	-
DMB	-	Data memory barrier	-
DSB	-	Data synchronization barrier	-
EOR, EORS	{Rd, } Rn , Op2	Exclusive OR	N,Z,C
ISB	-	Instruction synchronization barrier	-
IT	-	If-Then condition block	-
LDM	Rn{ ! } , reglist	Load multiple registers, increment after	-
LDMDB, LDMEA	Rn{ ! } , reglist	Load multiple registers, decrement before	-
LDMFD, LDMIA	Rn{ ! } , reglist	Load multiple registers, increment after	-
LDR	Rt , [Rn { , #offset }]	Load register with word	-
LDRB, LDRBT	Rt , [Rn { , #offset }]	Load register with byte	-
LDRD	Rt , Rt2 , [Rn { , #offset }]	Load register with two words	-

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

Mnemonic	Operands	Brief Description	Flags
LDREX	Rt , [Rn , #offset]	Load register exclusive	-
LDREXB	Rt , [Rn]	Load register exclusive with byte	-
LDREXH	Rt , [Rn]	Load register exclusive with halfword	-
LDRH, LDRHT	Rt , [Rn{ , #offset }]	Load register with halfword	-
LDRSB, LDRSBT	Rt , [Rn{ , #offset }]	Load register with signed byte	-
LDRSH, LDRSHT	Rt , [Rn{ , #offset }]	Load register with signed halfword	-
LDRT	Rt , [Rn { , #offset }]	Load register with word	-
LSL, LSLS	Rd , Rm , <Rs #n>	Logical shift left	N,Z,C
LSR, LSRS	Rd , Rm , <Rs #n>	Logical shift right	N,Z,C
MLA	Rd , Rn , Rm , Ra	Multiply with accumulate, 32-bit result	-
MLS	Rd , Rn , Rm , Ra	Multiply and subtract, 32-bit result	-
MOV, MOVS	Rd , Op2	Move	N,Z,C
MOV, MOVW	Rd , #imm16	Move 16-bit constant	N,Z,C
MOVT	Rd , #imm16	Move top	-
MRS	Rd , spec_reg	Move from special register to general register	-
MSR	spec_reg , R n	Move from general register to special register	N,Z,C,V
MUL, MULS	{Rd, } Rn , Rm	Multiply, 32-bit result	N,Z
MVN, MVNS	Rd , Op2	Move NOT	N,Z,C
NOP	-	No operation	-
ORN, ORNS	{Rd, } Rn , Op2	Logical OR NOT	N,Z,C
ORR, ORRS	{Rd, } Rn , Op2	Logical OR	N,Z,C
POP	reglist	Pop registers from stack	-
PUSH	reglist	Push registers onto stack	-
RBIT	Rd , Rn	Reverse bits	-
REV	Rd , Rn	Reverse byte order in a word	-
REV16	Rd , Rn	Reverse byte order in each halfword	-
REVSH	Rd , Rn	Reverse byte order in bottom halfword and sign extend	-
ROR, RORS	Rd , Rm , <Rs #n>	Rotate right	N,Z,C
RRX, RRXS	Rd , Rm	Rotate right with extend	N,Z,C
RSB, RSBS	{Rd, } Rn , Op2	Reverse subtract	N,Z,C,V
SBC, SBCS	{Rd, } Rn , Op2	Subtract with carry	N,Z,C,V
SBFX	Rd , Rn , #lsb , #width	Signed bit field extract	-
SDIV	{Rd, } Rn , Rm	Signed divide	-
SEV	-	Send event	-
SMLAL	RdLo, RdHi, Rn, Rm	Signed multiply with accumulate (32x32+64), 64-bit result	-
SMULL	RdLo, RdHi, Rn, Rm	Signed multiply (32x32), 64-bit result	-
SSAT	Rd, #n, Rm { ,shift #s }	Signed saturate	Q
STM	Rn{ ! } , reglist	Store multiple registers, increment after	-

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

Mnemonic	Operands	Brief Description	Flags
STMDB, STMEA	Rn{!} , reglist	Store multiple registers, decrement before	-
STMFD, STMIA	Rn{!} , reglist	Store multiple registers, increment after	-
STR	Rt , [Rn {, #offset}]	Store register word	-
STRB, STRBT	Rt , [Rn {, #offset}]	Store register byte	-
STRD	Rt , Rt2 , [Rn {, #offset}]	Store register two words	-
STREX	Rd , Rt , [Rn , #offset]	Store register exclusive	-
STREXB	Rd , Rt , [Rn]	Store register exclusive byte	-
STREXH	Rd , Rt , [Rn]	Store register exclusive halfword	-
STRH, STRHT	Rt , [Rn {, #offset}]	Store register halfword	-
STRSB, STRSBT	Rt , [Rn {, #offset}]	Store register signed byte	-
STRSH, STRSHT	Rt , [Rn {, #offset}]	Store register signed halfword	-
STRT	Rt , [Rn {, #offset}]	Store register word	-
SUB, SUBS	{Rd,} Rn , Op2	Subtract	N,Z,C,V
SUB, SUBW	{Rd,} Rn , #imm12	Subtract 12-bit constant	N,Z,C,V
SVC	#imm	Supervisor call	-
SXTB	{Rd,} Rm {,ROR #n}	Sign extend a byte	-
SXTH	{Rd,} Rm {,ROR #n}	Sign extend a halfword	-
TBB	[Rn, Rm]	Table branch byte	-
TBH	[Rn, Rm, LSL #1]	Table branch halfword	-
TEQ	Rn, Op2	Test equivalence	N,Z,C
TST	Rn, Op2	Test	N,Z,C
UBFX	Rd , Rn , #lsb , #width	Unsigned bit field extract	-
UDIV	{Rd,} Rn , Rm	Unsigned divide	-
UMLAL	RdLo, RdHi, Rn, Rm	Unsigned multiply with accumulate (32x32+64), 64-bit result	-
UMULL	RdLo, RdHi, Rn, Rm	Unsigned multiply (32x 2), 64-bit result	-
USAT	Rd, #n, Rm {,shift #s}	Unsigned saturate	Q
UXTB	{Rd,} Rm {,ROR #n}	Zero extend a byte	-
UXTH	{Rd,} Rm {,ROR #n}	Zero extend a halfword	-
WFE	-	Wait for event	-
WFI	-	Wait for interrupt	-

3 Cortex-M3 Peripherals

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

- SysTick (see page 116)

Provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism.
- Nested Vectored Interrupt Controller (NVIC) (see page 117)
 - Facilitates low-latency exception and interrupt handling
 - Controls power management
 - Implements system control registers
- System Control Block (SCB) (see page 119)

Provides system implementation information and system control, including configuration, control, and reporting of system exceptions.
- Memory Protection Unit (MPU) (see page 119)

Supports the standard ARMv7 Protected Memory System Architecture (PMSA) model. The MPU provides full support for protection regions, overlapping protection regions, access permissions, and exporting memory attributes to the system.

Table 3-1 on page 116 shows the address map of the Private Peripheral Bus (PPB). Some peripheral register regions are split into two address regions, as indicated by two addresses listed.

Table 3-1. Core Peripheral Register Regions

Address	Core Peripheral	Description (see page ...)
0xE000.E010-0xE000.E01F	System Timer	116
0xE000.E100-0xE000.E4EF 0xE000.EF00-0xE000.EF03	Nested Vectored Interrupt Controller	117
0xE000.E008-0xE000.E00F 0xE000.ED00-0xE000.ED3F	System Control Block	119
0xE000.ED90-0xE000.EDB8	Memory Protection Unit	119

3.1 Functional Description

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

3.1.1 System Timer (SysTick)

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

- An RTOS tick timer that fires at a programmable rate (for example, 100 Hz) and invokes a SysTick routine.
- A high-speed alarm timer using the system clock.

- A variable rate alarm or signal timer—the duration is range-dependent on the reference clock used and the dynamic range of the counter.
- A simple counter used to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNT bit in the **STCTRL** control and status register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

The timer consists of three registers:

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

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

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

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

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

3.1.2 Nested Vectored Interrupt Controller (NVIC)

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

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

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

3.1.2.1 Level-Sensitive and Pulse Interrupts

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

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

When the processor enters the ISR, it automatically removes the pending state from the interrupt (see “Hardware and Software Control of Interrupts” on page 118 for more information). For a level-sensitive interrupt, if the signal is not deasserted before the processor returns from the ISR, the interrupt becomes pending again, and the processor must execute its ISR again. As a result, the peripheral can hold the interrupt signal asserted until it no longer needs servicing.

3.1.2.2 Hardware and Software Control of Interrupts

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

- The NVIC detects that the interrupt signal is High and the interrupt is not active.
- The NVIC detects a rising edge on the interrupt signal.
- Software writes to the corresponding interrupt set-pending register bit, or to the **Software Trigger Interrupt (SWTRIG)** register to make a Software-Generated Interrupt pending. See the `INT` bit in the **PEND0** register on page 135 or **SWTRIG** on page 143.

A pending interrupt remains pending until one of the following:

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

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

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

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

3.1.3 System Control Block (SCB)

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

3.1.4 Memory Protection Unit (MPU)

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

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

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

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

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

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

Configuration of MPU regions is based on memory types (see “Memory Regions, Types and Attributes” on page 94 for more information).

Table 3-2 on page 119 shows the possible MPU region attributes. See the section called “MPU Configuration for a Stellaris Microcontroller” on page 123 for guidelines for programming a microcontroller implementation.

Table 3-2. Memory Attributes Summary

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

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

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

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

The processor does not support unaligned accesses to MPU registers.

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

3.1.4.1 Updating an MPU Region

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

Updating an MPU Region Using Separate Words

This example simple code configures one region:

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

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

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

Software must use memory barrier instructions:

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

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

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

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

Updating an MPU Region Using Multi-Word Writes

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

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

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

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

This operation can be done in two words for pre-packed information, meaning that the **MPU Region Base Address (MPUBASE)** register (see page 177) contains the required region number and has the `VALID` bit set. This method can be used when the data is statically packed, for example in a boot loader:

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

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

```
; R1 = address and region number in one
; R2 = size and attributes in one
LDR R0, =MPUBASE ; 0xE000ED9C, MPU Region Base register
STM R0, {R1-R2} ; Region base address, region number and VALID bit,
; and Region Attribute, Size and Enable
```

Subregions

Regions of 256 bytes or more are divided into eight equal-sized subregions. Set the corresponding bit in the `SRD` field of the **MPU Region Attribute and Size (MPUATTR)** register (see page 179) to disable a subregion. The least-significant bit of the `SRD` field controls the first subregion, and the most-significant bit controls the last subregion. Disabling a subregion means another region

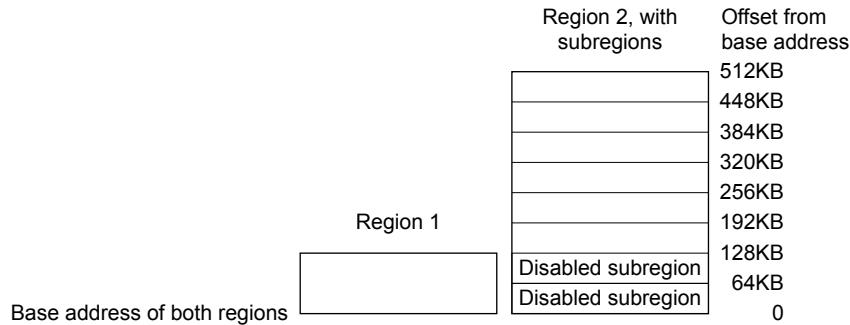
overlapping the disabled range matches instead. If no other enabled region overlaps the disabled subregion, the MPU issues a fault.

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

Example of SRD Use

Two regions with the same base address overlap. Region one is 128 KB, and region two is 512 KB. To ensure the attributes from region one apply to the first 128 KB region, configure the SRD field for region two to 0x03 to disable the first two subregions, as Figure 3-1 on page 122 shows.

Figure 3-1. SRD Use Example



3.1.4.2 MPU Access Permission Attributes

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

Table 3-3 on page 122 shows the encodings for the TEX, C, B, and S access permission bits. All encodings are shown for completeness, however the current implementation of the Cortex-M3 does not support the concept of cacheability or shareability. Refer to the section called “MPU Configuration for a Stellaris Microcontroller” on page 123 for information on programming the MPU for Stellaris implementations.

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

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

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

TEX	S	C	B	Memory Type	Shareability	Other Attributes
010	x ^a	0	0	Device	Not shareable	Nonshared Device.
010	x ^a	0	1	Reserved encoding	-	-
010	x ^a	1	x ^a	Reserved encoding	-	-
1BB	0	A	A	Normal	Not shareable	Cached memory (BB = outer policy, AA = inner policy).
1BB	1	A	A	Normal	Shareable	See Table 3-4 for the encoding of the AA and BB bits.

a. The MPU ignores the value of this bit.

Table 3-4 on page 123 shows the cache policy for memory attribute encodings with a **TEX** value in the range of 0x4-0x7.

Table 3-4. Cache Policy for Memory Attribute Encoding

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

Table 3-5 on page 123 shows the **AP** encodings in the **MPUATTR** register that define the access permissions for privileged and unprivileged software.

Table 3-5. AP Bit Field Encoding

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

MPU Configuration for a Stellaris Microcontroller

Stellaris microcontrollers have only a single processor and no caches. As a result, the MPU should be programmed as shown in Table 3-6 on page 123.

Table 3-6. Memory Region Attributes for Stellaris Microcontrollers

Memory Region	TEX	S	C	B	Memory Type and Attributes
Flash memory	000b	0	1	0	Normal memory, non-shareable, write-through
Internal SRAM	000b	1	1	0	Normal memory, shareable, write-through

Table 3-6. Memory Region Attributes for Stellaris Microcontrollers (continued)

Memory Region	TEX	S	C	B	Memory Type and Attributes
External SRAM	000b	1	1	1	Normal memory, shareable, write-back, write-allocate
Peripherals	000b	1	0	1	Device memory, shareable

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

3.1.4.3 MPU Mismatch

When an access violates the MPU permissions, the processor generates a memory management fault (see “Exceptions and Interrupts” on page 92 for more information). The **MFAULTSTAT** register indicates the cause of the fault. See page 164 for more information.

3.2 Register Map

Table 3-7 on page 124 lists the Cortex-M3 Peripheral SysTick, NVIC, SCB, and MPU registers. The offset listed is a hexadecimal increment to the register's address, relative to the Core Peripherals base address of 0xE000.E000.

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

Table 3-7. Peripherals Register Map

Offset	Name	Type	Reset	Description	See page
System Timer (SysTick) Registers					
0x010	STCTRL	R/W	0x0000.0004	SysTick Control and Status Register	127
0x014	STRELOAD	R/W	0x0000.0000	SysTick Reload Value Register	129
0x018	STCURRENT	R/WC	0x0000.0000	SysTick Current Value Register	130
Nested Vectored Interrupt Controller (NVIC) Registers					
0x100	EN0	R/W	0x0000.0000	Interrupt 0-31 Set Enable	131
0x104	EN1	R/W	0x0000.0000	Interrupt 32-54 Set Enable	132
0x180	DIS0	R/W	0x0000.0000	Interrupt 0-31 Clear Enable	133
0x184	DIS1	R/W	0x0000.0000	Interrupt 32-54 Clear Enable	134
0x200	PEND0	R/W	0x0000.0000	Interrupt 0-31 Set Pending	135
0x204	PEND1	R/W	0x0000.0000	Interrupt 32-54 Set Pending	136
0x280	UNPEND0	R/W	0x0000.0000	Interrupt 0-31 Clear Pending	137
0x284	UNPEND1	R/W	0x0000.0000	Interrupt 32-54 Clear Pending	138
0x300	ACTIVE0	RO	0x0000.0000	Interrupt 0-31 Active Bit	139
0x304	ACTIVE1	RO	0x0000.0000	Interrupt 32-54 Active Bit	140
0x400	PRI0	R/W	0x0000.0000	Interrupt 0-3 Priority	141

Table 3-7. Peripherals Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x404	PRI1	R/W	0x0000.0000	Interrupt 4-7 Priority	141
0x408	PRI2	R/W	0x0000.0000	Interrupt 8-11 Priority	141
0x40C	PRI3	R/W	0x0000.0000	Interrupt 12-15 Priority	141
0x410	PRI4	R/W	0x0000.0000	Interrupt 16-19 Priority	141
0x414	PRI5	R/W	0x0000.0000	Interrupt 20-23 Priority	141
0x418	PRI6	R/W	0x0000.0000	Interrupt 24-27 Priority	141
0x41C	PRI7	R/W	0x0000.0000	Interrupt 28-31 Priority	141
0x420	PRI8	R/W	0x0000.0000	Interrupt 32-35 Priority	141
0x424	PRI9	R/W	0x0000.0000	Interrupt 36-39 Priority	141
0x428	PRI10	R/W	0x0000.0000	Interrupt 40-43 Priority	141
0x42C	PRI11	R/W	0x0000.0000	Interrupt 44-47 Priority	141
0x430	PRI12	R/W	0x0000.0000	Interrupt 48-51 Priority	141
0x434	PRI13	R/W	0x0000.0000	Interrupt 52-54 Priority	141
0xF00	SWTRIG	WO	0x0000.0000	Software Trigger Interrupt	143
System Control Block (SCB) Registers					
0x008	ACTLR	R/W	0x0000.0000	Auxiliary Control	144
0xD00	CPUID	RO	0x412F.C230	CPU ID Base	146
0xD04	INTCTRL	R/W	0x0000.0000	Interrupt Control and State	147
0xD08	VTABLE	R/W	0x0000.0000	Vector Table Offset	150
0xD0C	APINT	R/W	0xFA05.0000	Application Interrupt and Reset Control	151
0xD10	SYSCTRL	R/W	0x0000.0000	System Control	153
0xD14	CFGCTRL	R/W	0x0000.0200	Configuration and Control	155
0xD18	SYSPRI1	R/W	0x0000.0000	System Handler Priority 1	157
0xD1C	SYSPRI2	R/W	0x0000.0000	System Handler Priority 2	158
0xD20	SYSPRI3	R/W	0x0000.0000	System Handler Priority 3	159
0xD24	SYSHNDCTRL	R/W	0x0000.0000	System Handler Control and State	160
0xD28	FAULTSTAT	R/W1C	0x0000.0000	Configurable Fault Status	164
0xD2C	HFAULTSTAT	R/W1C	0x0000.0000	Hard Fault Status	170
0xD34	MMADDR	R/W	-	Memory Management Fault Address	171
0xD38	FAULTADDR	R/W	-	Bus Fault Address	172
Memory Protection Unit (MPU) Registers					
0xD90	MPUTYPE	RO	0x0000.0800	MPU Type	173

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

Offset	Name	Type	Reset	Description	See page
0xD94	MPUCTRL	R/W	0x0000.0000	MPU Control	174
0xD98	MPUNUMBER	R/W	0x0000.0000	MPU Region Number	176
0xD9C	MPUBASE	R/W	0x0000.0000	MPU Region Base Address	177
0xDA0	MPUATTR	R/W	0x0000.0000	MPU Region Attribute and Size	179
0xDA4	MPUBASE1	R/W	0x0000.0000	MPU Region Base Address Alias 1	177
0xDA8	MPUATTR1	R/W	0x0000.0000	MPU Region Attribute and Size Alias 1	179
0xDAC	MPUBASE2	R/W	0x0000.0000	MPU Region Base Address Alias 2	177
0xDB0	MPUATTR2	R/W	0x0000.0000	MPU Region Attribute and Size Alias 2	179
0xDB4	MPUBASE3	R/W	0x0000.0000	MPU Region Base Address Alias 3	177
0xDB8	MPUATTR3	R/W	0x0000.0000	MPU Region Attribute and Size Alias 3	179

3.3 System Timer (SysTick) Register Descriptions

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

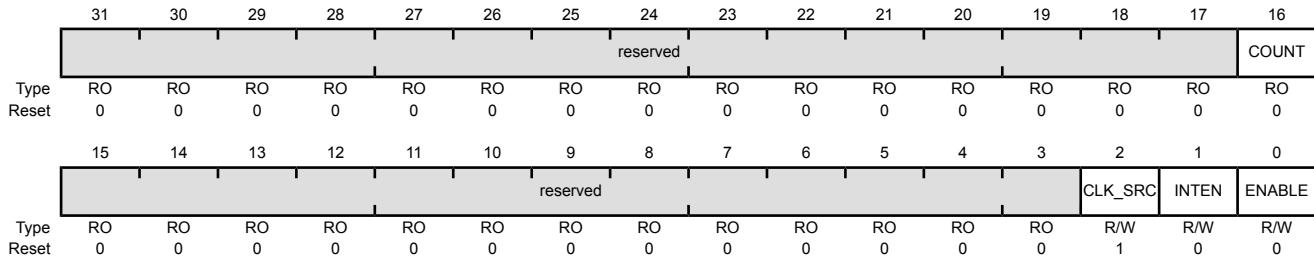
Register 1: SysTick Control and Status Register (STCTRL), offset 0x010

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

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

SysTick Control and Status Register (STCTRL)

Base 0xE000.E000
Offset 0x010
Type R/W, reset 0x0000.0004



Bit/Field	Name	Type	Reset	Description						
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
16	COUNT	RO	0	<p>Count Flag</p> <table border="1"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>The SysTick timer has not counted to 0 since the last time this bit was read.</td> </tr> <tr> <td>1</td> <td>The SysTick timer has counted to 0 since the last time this bit was read.</td> </tr> </table> <p>This bit is cleared by a read of the register or if the STCURRENT register is written with any value. 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 COUNT bit is not changed by the debugger read. See the <i>ARM® Debug Interface V5 Architecture Specification</i> for more information on MasterType.</p>	Value	Description	0	The SysTick timer has not counted to 0 since the last time this bit was read.	1	The SysTick timer has counted to 0 since the last time this bit was read.
Value	Description									
0	The SysTick timer has not counted to 0 since the last time this bit was read.									
1	The SysTick timer has counted to 0 since the last time this bit was read.									
15:3	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
2	CLK_SRC	R/W	1	<p>Clock Source</p> <table border="1"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>0</td> <td>External reference clock. (Not implemented for Stellaris microcontrollers.)</td> </tr> <tr> <td>1</td> <td>System clock</td> </tr> </table> <p>Because an external reference clock is not implemented, this bit must be set in order for SysTick to operate.</p>	Value	Description	0	External reference clock. (Not implemented for Stellaris microcontrollers.)	1	System clock
Value	Description									
0	External reference clock. (Not implemented for Stellaris microcontrollers.)									
1	System clock									

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

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

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

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

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

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

SysTick Reload Value Register (STRELOAD)

Base 0xE000.E000

Offset 0x014

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved														RELOAD	
Type	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
	RELOAD															
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
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:0	RELOAD	R/W	0x00.0000	Reload Value Value to load into the SysTick Current Value (STCURRENT) register when the counter reaches 0.

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

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

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

SysTick Current Value Register (STCURRENT)

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

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

Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:0	CURRENT	R/WC	0x00.0000	<p>Current Value</p> <p>This field contains the current value at the time the register is accessed. No read-modify-write protection is provided, so change with care.</p> <p>This register is write-clear. Writing to it with any value clears the register. Clearing this register also clears the COUNT bit of the STCTRL register.</p>

3.4 NVIC Register Descriptions

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

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

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

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

Before programming the **VTABLE** register to relocate the vector table, ensure the vector table entries of the new vector table are set up for fault handlers, NMI, and all enabled exceptions such as interrupts. For more information, see page 150.

Register 4: Interrupt 0-31 Set Enable (EN0), offset 0x100

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

The **EN0** register enables interrupts and shows which interrupts are enabled. Bit 0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31.

See Table 2-9 on page 104 for interrupt assignments.

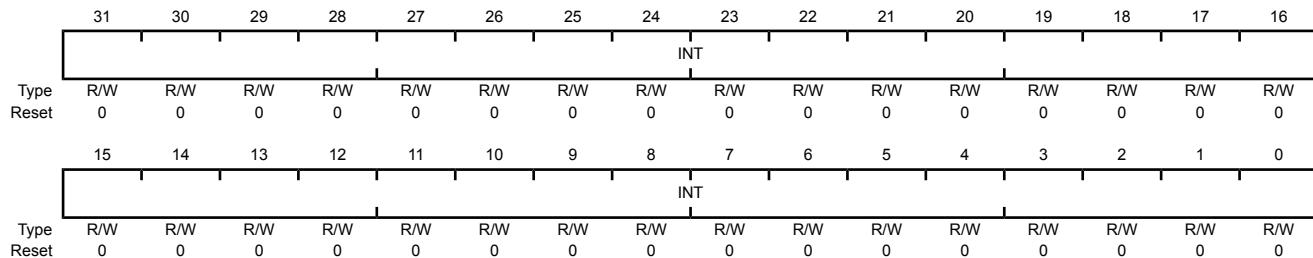
If a pending interrupt is enabled, the NVIC activates the interrupt based on its priority. If an interrupt is not enabled, asserting its interrupt signal changes the interrupt state to pending, but the NVIC never activates the interrupt, regardless of its priority.

Interrupt 0-31 Set Enable (EN0)

Base 0xE000.E000

Offset 0x100

Type R/W, reset 0x0000.0000



Bit/Field Name Type Reset Description

31:0	INT	R/W	0x0000.0000	Interrupt Enable
------	-----	-----	-------------	------------------

Value	Description
0	On a read, indicates the interrupt is disabled. On a write, no effect.
1	On a read, indicates the interrupt is enabled. On a write, enables the interrupt.

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

Register 5: Interrupt 32-54 Set Enable (EN1), offset 0x104

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

The **EN1** register enables interrupts and shows which interrupts are enabled. Bit 0 corresponds to Interrupt 32; bit 22 corresponds to Interrupt 54. See Table 2-9 on page 104 for interrupt assignments.

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

Interrupt 32-54 Set Enable (EN1)

Base 0xE000.E000

Offset 0x104

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	R/W														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
INT																
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
INT																

Bit/Field	Name	Type	Reset	Description
31:23	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22:0	INT	R/W	0x00.0000	Interrupt Enable

Value	Description
0	On a read, indicates the interrupt is disabled. On a write, no effect.
1	On a read, indicates the interrupt is enabled. On a write, enables the interrupt.

A bit can only be cleared by setting the corresponding `INT[n]` bit in the **DIS1** register.

Register 6: Interrupt 0-31 Clear Enable (DIS0), offset 0x180

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

The **DIS0** register disables interrupts. Bit 0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31.

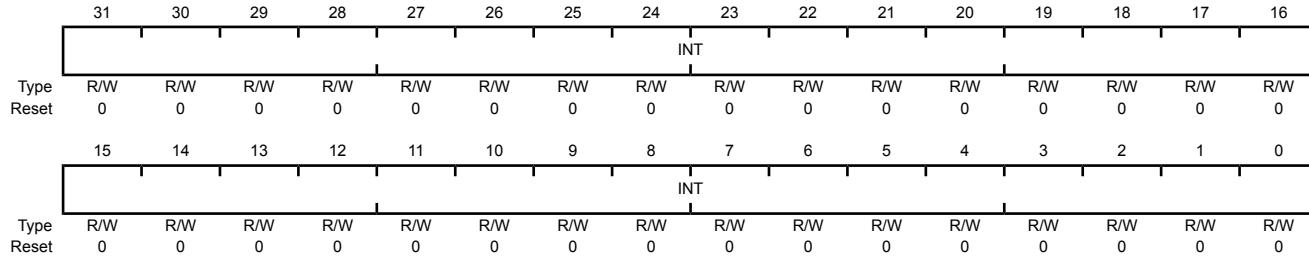
See Table 2-9 on page 104 for interrupt assignments.

Interrupt 0-31 Clear Enable (DIS0)

Base 0xE000.E000

Offset 0x180

Type R/W, reset 0x0000.0000



Bit/Field Name Type Reset Description

31:0 INT R/W 0x0000.0000 Interrupt Disable

Value	Description
0	On a read, indicates the interrupt is disabled. On a write, no effect.
1	On a read, indicates the interrupt is enabled. On a write, clears the corresponding <code>INT[n]</code> bit in the EN0 register, disabling interrupt [n].

Register 7: Interrupt 32-54 Clear Enable (DIS1), offset 0x184

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

The **DIS1** register disables interrupts. Bit 0 corresponds to Interrupt 32; bit 22 corresponds to Interrupt 54. See Table 2-9 on page 104 for interrupt assignments.

Interrupt 32-54 Clear Enable (DIS1)

Base 0xE000.E000
Offset 0x184
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	R/W														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
INT																
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field Name Type Reset Description

31:23 reserved RO 0x00 Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

22:0 INT R/W 0x00.0000 Interrupt Disable

Value	Description
0	On a read, indicates the interrupt is disabled. On a write, no effect.
1	On a read, indicates the interrupt is enabled. On a write, clears the corresponding <code>INT[n]</code> bit in the EN1 register, disabling interrupt [n].

Register 8: Interrupt 0-31 Set Pending (PEND0), offset 0x200

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

The **PEND0** register forces interrupts into the pending state and shows which interrupts are pending. Bit 0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31.

See Table 2-9 on page 104 for interrupt assignments.

Interrupt 0-31 Set Pending (PEND0)

Base 0xE000.E000

Offset 0x200

Type R/W, reset 0x0000.0000

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

31:0 INT R/W 0x0000.0000 Interrupt Set Pending

Value	Description
0	On a read, indicates that the interrupt is not pending. On a write, no effect.
1	On a read, indicates that the interrupt is pending. On a write, the corresponding interrupt is set to pending even if it is disabled.

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

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

Register 9: Interrupt 32-54 Set Pending (PEND1), offset 0x204

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

The **PEND1** register forces interrupts into the pending state and shows which interrupts are pending. Bit 0 corresponds to Interrupt 32; bit 22 corresponds to Interrupt 54. See Table 2-9 on page 104 for interrupt assignments.

Interrupt 32-54 Set Pending (PEND1)

Base 0xE000.E000
Offset 0x204
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	R/W														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
INT																
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
INT																
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:23	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22:0	INT	R/W	0x00.0000	Interrupt Set Pending
		Value		Description
		0		On a read, indicates that the interrupt is not pending. On a write, no effect.
		1		On a read, indicates that the interrupt is pending. On a write, the corresponding interrupt is set to pending even if it is disabled.
				If the corresponding interrupt is already pending, setting a bit has no effect.
				A bit can only be cleared by setting the corresponding INT[n] bit in the UNPEND1 register.

Register 10: Interrupt 0-31 Clear Pending (UNPEND0), offset 0x280

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

The **UNPEND0** register shows which interrupts are pending and removes the pending state from interrupts. Bit 0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31.

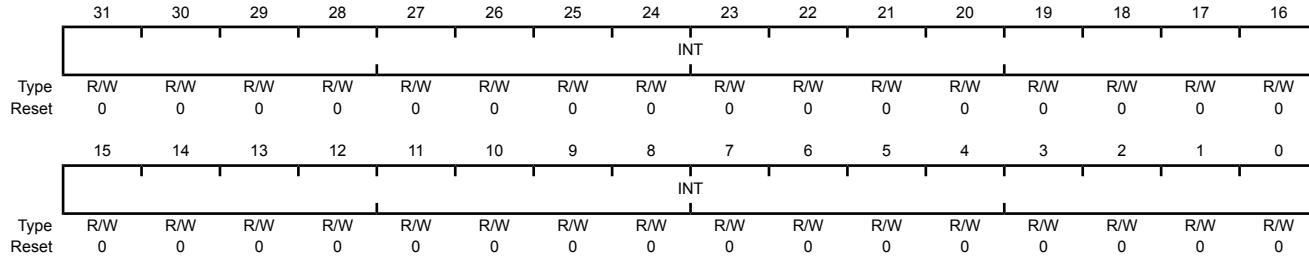
See Table 2-9 on page 104 for interrupt assignments.

Interrupt 0-31 Clear Pending (UNPEND0)

Base 0xE000.E000

Offset 0x280

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	INT	R/W	0x0000.0000	Interrupt Clear Pending
Value Description				
0				On a read, indicates that the interrupt is not pending. On a write, no effect.
1				On a read, indicates that the interrupt is pending. On a write, clears the corresponding <code>INT[n]</code> bit in the PEND0 register, so that interrupt [n] is no longer pending. Setting a bit does not affect the active state of the corresponding interrupt.

Register 11: Interrupt 32-54 Clear Pending (UNPEND1), offset 0x284

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

The **UNPEND1** register shows which interrupts are pending and removes the pending state from interrupts. Bit 0 corresponds to Interrupt 32; bit 22 corresponds to Interrupt 54. See Table 2-9 on page 104 for interrupt assignments.

Interrupt 32-54 Clear Pending (UNPEND1)

Base 0xE000.E000
Offset 0x284
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	R/W														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
INT																
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
INT																
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:23	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22:0	INT	R/W	0x00.0000	Interrupt Clear Pending
		Value	Description	
		0	On a read, indicates that the interrupt is not pending. On a write, no effect.	
		1	On a read, indicates that the interrupt is pending. On a write, clears the corresponding INT[n] bit in the PEND1 register, so that interrupt [n] is no longer pending. Setting a bit does not affect the active state of the corresponding interrupt.	

Register 12: Interrupt 0-31 Active Bit (ACTIVE0), offset 0x300

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

The **ACTIVE0** register indicates which interrupts are active. Bit 0 corresponds to Interrupt 0; bit 31 corresponds to Interrupt 31.

See Table 2-9 on page 104 for interrupt assignments.

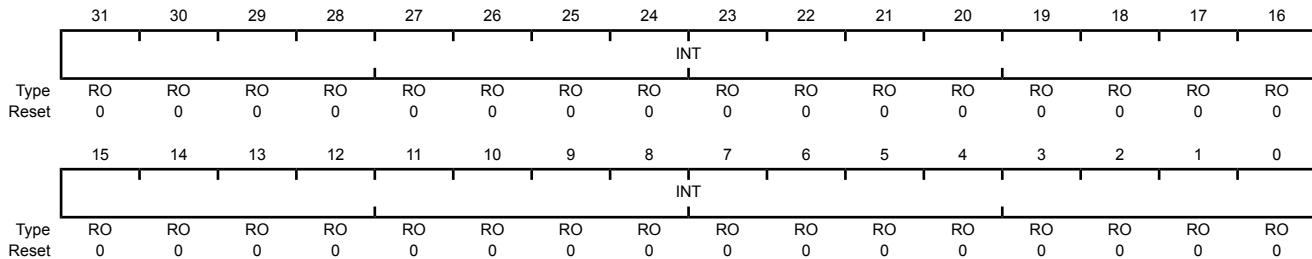
Caution – Do not manually set or clear the bits in this register.

Interrupt 0-31 Active Bit (ACTIVE0)

Base 0xE000.E000

Offset 0x300

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	INT	RO	0x0000.0000	Interrupt Active

Value Description

0 The corresponding interrupt is not active.

1 The corresponding interrupt is active, or active and pending.

Register 13: Interrupt 32-54 Active Bit (ACTIVE1), offset 0x304

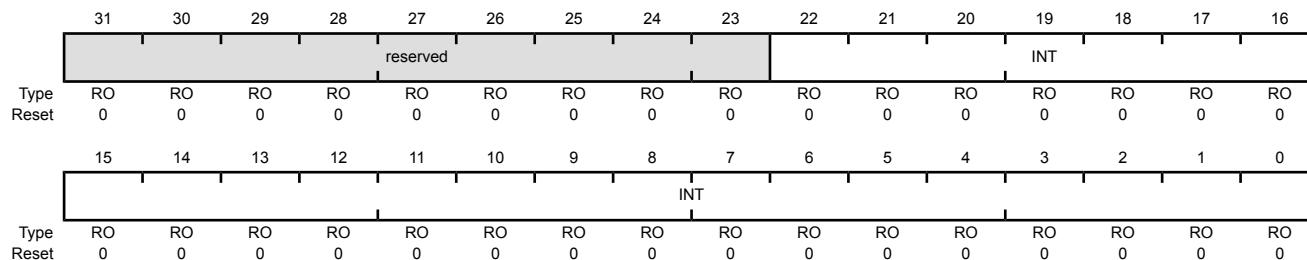
Note: This register can only be accessed from privileged mode.

The **ACTIVE1** register indicates which interrupts are active. Bit 0 corresponds to Interrupt 32; bit 22 corresponds to Interrupt 54. See Table 2-9 on page 104 for interrupt assignments.

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

Interrupt 32-54 Active Bit (ACTIVE1)

Base 0xE000.E000
Offset 0x304
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:23	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22:0	INT	RO	0x00.0000	Interrupt Active
		Value	Description	
		0	The corresponding interrupt is not active.	
		1	The corresponding interrupt is active, or active and pending.	

- Register 14: Interrupt 0-3 Priority (PRI0), offset 0x400**
- Register 15: Interrupt 4-7 Priority (PRI1), offset 0x404**
- Register 16: Interrupt 8-11 Priority (PRI2), offset 0x408**
- Register 17: Interrupt 12-15 Priority (PRI3), offset 0x40C**
- Register 18: Interrupt 16-19 Priority (PRI4), offset 0x410**
- Register 19: Interrupt 20-23 Priority (PRI5), offset 0x414**
- Register 20: Interrupt 24-27 Priority (PRI6), offset 0x418**
- Register 21: Interrupt 28-31 Priority (PRI7), offset 0x41C**
- Register 22: Interrupt 32-35 Priority (PRI8), offset 0x420**
- Register 23: Interrupt 36-39 Priority (PRI9), offset 0x424**
- Register 24: Interrupt 40-43 Priority (PRI10), offset 0x428**
- Register 25: Interrupt 44-47 Priority (PRI11), offset 0x42C**
- Register 26: Interrupt 48-51 Priority (PRI12), offset 0x430**
- Register 27: Interrupt 52-54 Priority (PRI13), offset 0x434**

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

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

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

See Table 2-9 on page 104 for interrupt assignments.

Each priority level can be split into separate group priority and subpriority fields. The PRIGROUP field in the **Application Interrupt and Reset Control (APINT)** register (see page 151) indicates the position of the binary point that splits the priority and subpriority fields .

These registers can only be accessed from privileged mode.

Interrupt 0-3 Priority (PRI0)

Base 0xE000.E000
Offset 0x400
Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:29	INTD	R/W	0x0	Interrupt Priority for Interrupt [4n+3] This field holds a priority value, 0-7, for the interrupt with the number [4n+3], where n is the number of the Interrupt Priority register (n=0 for PRI0 , and so on). The lower the value, the greater the priority of the corresponding interrupt.
28:24	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:21	INTC	R/W	0x0	Interrupt Priority for Interrupt [4n+2] This field holds a priority value, 0-7, for the interrupt with the number [4n+2], where n is the number of the Interrupt Priority register (n=0 for PRI0 , and so on). The lower the value, the greater the priority of the corresponding interrupt.
20:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:13	INTB	R/W	0x0	Interrupt Priority for Interrupt [4n+1] This field holds a priority value, 0-7, for the interrupt with the number [4n+1], where n is the number of the Interrupt Priority register (n=0 for PRI0 , and so on). The lower the value, the greater the priority of the corresponding interrupt.
12:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	INTA	R/W	0x0	Interrupt Priority for Interrupt [4n] This field holds a priority value, 0-7, for the interrupt with the number [4n], where n is the number of the Interrupt Priority register (n=0 for PRI0 , and so on). The lower the value, the greater the priority of the corresponding interrupt.
4:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 28: Software Trigger Interrupt (SWTRIG), offset 0xF00

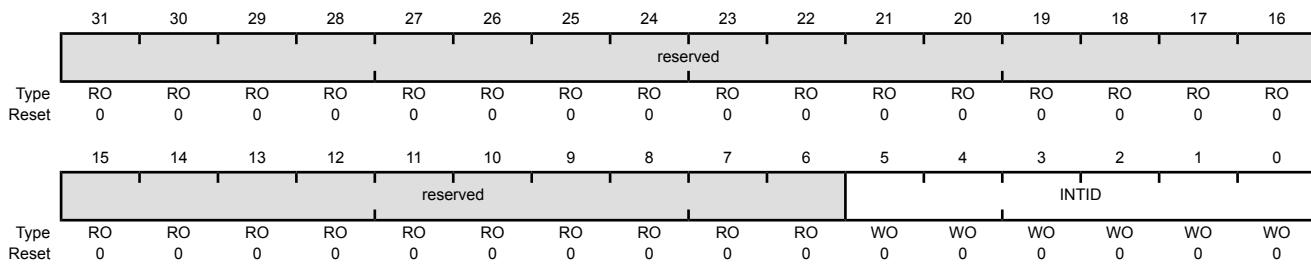
Note: Only privileged software can enable unprivileged access to the **SWTRIG** register.

Writing an interrupt number to the **SWTRIG** register generates a Software Generated Interrupt (SGI). See Table 2-9 on page 104 for interrupt assignments.

When the **MAINPEND** bit in the **Configuration and Control (CFGCTRL)** register (see page 155) is set, unprivileged software can access the **SWTRIG** register.

Software Trigger Interrupt (SWTRIG)

Base 0xE000.E000
Offset 0xF00
Type WO, reset 0x0000.0000



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.
5:0	INTID	WO	0x00	Interrupt ID This field holds the interrupt ID of the required SGI. For example, a value of 0x3 generates an interrupt on IRQ3.

3.5 System Control Block (SCB) Register Descriptions

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

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

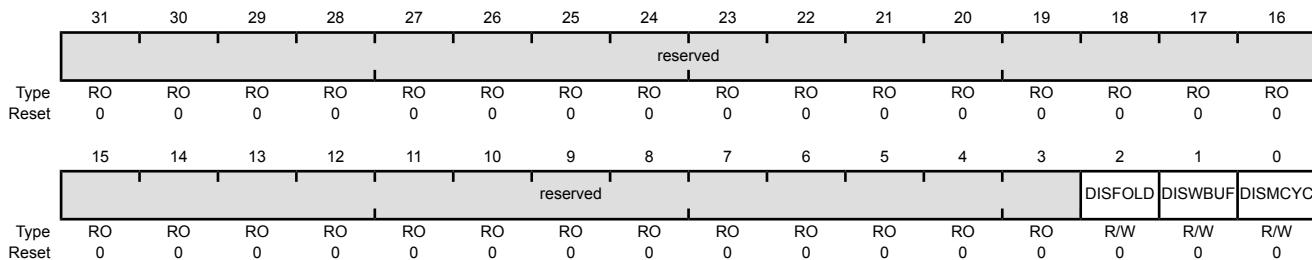
Register 29: Auxiliary Control (ACTLR), offset 0x008

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

The **ACTLR** register provides disable bits for **IT** folding, write buffer use for accesses to the default memory map, and interruption of multi-cycle instructions. By default, this register is set to provide optimum performance from the Cortex-M3 processor and does not normally require modification.

Auxiliary Control (ACTLR)

Base 0xE000.E000
Offset 0x008
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	DISFOLD	R/W	0	<p>Disable IT Folding</p> <p>Value Description</p> <p>0 No effect.</p> <p>1 Disables IT folding.</p> <p>In some situations, the processor can start executing the first instruction in an IT block while it is still executing the IT instruction. This behavior is called IT folding, and improves performance. However, IT folding can cause jitter in looping. If a task must avoid jitter, set the DISFOLD bit before executing the task, to disable IT folding.</p>
1	DISWBUF	R/W	0	<p>Disable Write Buffer</p> <p>Value Description</p> <p>0 No effect.</p> <p>1 Disables write buffer use during default memory map accesses. In this situation, all bus faults are precise bus faults but performance is decreased because any store to memory must complete before the processor can execute the next instruction.</p> <p>Note: This bit only affects write buffers implemented in the Cortex-M3 processor.</p>

Bit/Field	Name	Type	Reset	Description
0	DISMCYC	R/W	0	Disable Interrupts of Multiple Cycle Instructions
				Value Description
			0	No effect.
			1	Disables interruption of load multiple and store multiple instructions. In this situation, the interrupt latency of the processor is increased because any LDM or STM must complete before the processor can stack the current state and enter the interrupt handler.

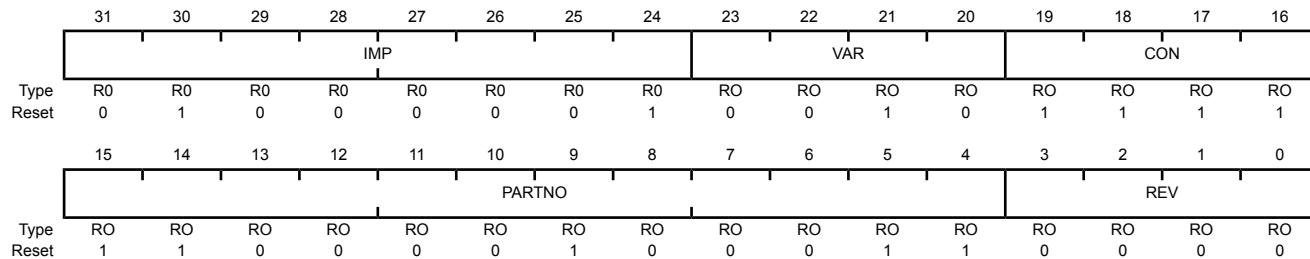
Register 30: CPU ID Base (CPUID), offset 0xD00

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

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

CPU ID Base (CPUID)

Base 0xE000.E000
Offset 0xD00
Type RO, reset 0x412F.C230



Bit/Field	Name	Type	Reset	Description
31:24	IMP	RO	0x41	Implementer Code Value Description 0x41 ARM
23:20	VAR	RO	0x2	Variant Number Value Description 0x2 The rn value in the rnpn product revision identifier, for example, the 2 in r2p0.
19:16	CON	RO	0xF	Constant Value Description 0xF Always reads as 0xF.
15:4	PARTNO	RO	0xC23	Part Number Value Description 0xC23 Cortex-M3 processor.
3:0	REV	RO	0x0	Revision Number Value Description 0x0 The pn value in the rnpn product revision identifier, for example, the 0 in r2p0.

Register 31: Interrupt Control and State (INTCTRL), offset 0xD04

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

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

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

Interrupt Control and State (INTCTRL)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NMISET	reserved		PENDSV	UNPENDSV	PENDSTSET	PENDSTCLR	reserved	ISRPRE	ISRPEND		reserved				VECPEND
Type	R/W	RO	RO	R/W	WO	R/W	WO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	VECPEND			RETBASE		reserved							VECACT			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field Name Type Reset Description

31 NMISET R/W 0 NMI Set Pending

Value Description

- 0 On a read, indicates an NMI exception is not pending.
On a write, no effect.
- 1 On a read, indicates an NMI exception is pending.
On a write, changes the NMI exception state to pending.

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

30:29 reserved RO 0x0 Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

28 PENDSV R/W 0 PendSV Set Pending

Value Description

- 0 On a read, indicates a PendSV exception is not pending.
On a write, no effect.
- 1 On a read, indicates a PendSV exception is pending.
On a write, changes the PendSV exception state to pending.

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

Bit/Field	Name	Type	Reset	Description						
27	UNPENDSV	WO	0	<p>PendSV Clear Pending</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>On a write, no effect.</td></tr> <tr> <td>1</td><td>On a write, removes the pending state from the PendSV exception.</td></tr> </tbody> </table> <p>This bit is write only; on a register read, its value is unknown.</p>	Value	Description	0	On a write, no effect.	1	On a write, removes the pending state from the PendSV exception.
Value	Description									
0	On a write, no effect.									
1	On a write, removes the pending state from the PendSV exception.									
26	PENDSTSET	R/W	0	<p>SysTick Set Pending</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>On a read, indicates a SysTick exception is not pending. On a write, no effect.</td></tr> <tr> <td>1</td><td>On a read, indicates a SysTick exception is pending. On a write, changes the SysTick exception state to pending.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the PENDSTCLR bit.</p>	Value	Description	0	On a read, indicates a SysTick exception is not pending. On a write, no effect.	1	On a read, indicates a SysTick exception is pending. On a write, changes the SysTick exception state to pending.
Value	Description									
0	On a read, indicates a SysTick exception is not pending. On a write, no effect.									
1	On a read, indicates a SysTick exception is pending. On a write, changes the SysTick exception state to pending.									
25	PENDSTCLR	WO	0	<p>SysTick Clear Pending</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>On a write, no effect.</td></tr> <tr> <td>1</td><td>On a write, removes the pending state from the SysTick exception.</td></tr> </tbody> </table> <p>This bit is write only; on a register read, its value is unknown.</p>	Value	Description	0	On a write, no effect.	1	On a write, removes the pending state from the SysTick exception.
Value	Description									
0	On a write, no effect.									
1	On a write, removes the pending state from the SysTick exception.									
24	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
23	ISRPRE	RO	0	<p>Debug Interrupt Handling</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The release from halt does not take an interrupt.</td></tr> <tr> <td>1</td><td>The release from halt takes an interrupt.</td></tr> </tbody> </table> <p>This bit is only meaningful in Debug mode and reads as zero when the processor is not in Debug mode.</p>	Value	Description	0	The release from halt does not take an interrupt.	1	The release from halt takes an interrupt.
Value	Description									
0	The release from halt does not take an interrupt.									
1	The release from halt takes an interrupt.									
22	ISRPEND	RO	0	<p>Interrupt Pending</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No interrupt is pending.</td></tr> <tr> <td>1</td><td>An interrupt is pending.</td></tr> </tbody> </table> <p>This bit provides status for all interrupts excluding NMI and Faults.</p>	Value	Description	0	No interrupt is pending.	1	An interrupt is pending.
Value	Description									
0	No interrupt is pending.									
1	An interrupt is pending.									
21:19	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																																						
18:12	VECPEND	RO	0x00	<p>Interrupt Pending Vector Number</p> <p>This field contains the exception number of the highest priority pending enabled exception. The value indicated by this field includes the effect of the BASEPRI and FAULTMASK registers, but not any effect of the PRIMASK register.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr><td>0x00</td><td>No exceptions are pending</td></tr> <tr><td>0x01</td><td>Reserved</td></tr> <tr><td>0x02</td><td>NMI</td></tr> <tr><td>0x03</td><td>Hard fault</td></tr> <tr><td>0x04</td><td>Memory management fault</td></tr> <tr><td>0x05</td><td>Bus fault</td></tr> <tr><td>0x06</td><td>Usage fault</td></tr> <tr><td>0x07-0x0A</td><td>Reserved</td></tr> <tr><td>0x0B</td><td>SVCALL</td></tr> <tr><td>0x0C</td><td>Reserved for Debug</td></tr> <tr><td>0x0D</td><td>Reserved</td></tr> <tr><td>0x0E</td><td>PendSV</td></tr> <tr><td>0x0F</td><td>SysTick</td></tr> <tr><td>0x10</td><td>Interrupt Vector 0</td></tr> <tr><td>0x11</td><td>Interrupt Vector 1</td></tr> <tr><td>...</td><td>...</td></tr> <tr><td>0x46</td><td>Interrupt Vector 54</td></tr> <tr><td>0x47-0x7F</td><td>Reserved</td></tr> </tbody> </table>	Value	Description	0x00	No exceptions are pending	0x01	Reserved	0x02	NMI	0x03	Hard fault	0x04	Memory management fault	0x05	Bus fault	0x06	Usage fault	0x07-0x0A	Reserved	0x0B	SVCALL	0x0C	Reserved for Debug	0x0D	Reserved	0x0E	PendSV	0x0F	SysTick	0x10	Interrupt Vector 0	0x11	Interrupt Vector 1	0x46	Interrupt Vector 54	0x47-0x7F	Reserved
Value	Description																																									
0x00	No exceptions are pending																																									
0x01	Reserved																																									
0x02	NMI																																									
0x03	Hard fault																																									
0x04	Memory management fault																																									
0x05	Bus fault																																									
0x06	Usage fault																																									
0x07-0x0A	Reserved																																									
0x0B	SVCALL																																									
0x0C	Reserved for Debug																																									
0x0D	Reserved																																									
0x0E	PendSV																																									
0x0F	SysTick																																									
0x10	Interrupt Vector 0																																									
0x11	Interrupt Vector 1																																									
...	...																																									
0x46	Interrupt Vector 54																																									
0x47-0x7F	Reserved																																									
11	RETBASE	RO	0	<p>Return to Base</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr><td>0</td><td>There are preempted active exceptions to execute.</td></tr> <tr><td>1</td><td>There are no active exceptions, or the currently executing exception is the only active exception.</td></tr> </tbody> </table> <p>This bit provides status for all interrupts excluding NMI and Faults. This bit only has meaning if the processor is currently executing an ISR (the Interrupt Program Status (IPSR) register is non-zero).</p>	Value	Description	0	There are preempted active exceptions to execute.	1	There are no active exceptions, or the currently executing exception is the only active exception.																																
Value	Description																																									
0	There are preempted active exceptions to execute.																																									
1	There are no active exceptions, or the currently executing exception is the only active exception.																																									
10:7	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.																																						
6:0	VECACT	RO	0x00	<p>Interrupt Pending Vector Number</p> <p>This field contains the active exception number. The exception numbers can be found in the description for the VECPEND field. If this field is clear, the processor is in Thread mode. This field contains the same value as the ISRNUM field in the IPSR register.</p> <p>Subtract 16 from this value to obtain the IRQ number required to index into the Interrupt Set Enable (ENn), Interrupt Clear Enable (DISn), Interrupt Set Pending (PENDn), Interrupt Clear Pending (UNPENDn), and Interrupt Priority (PRIIn) registers (see page 84).</p>																																						

Register 32: Vector Table Offset (VTABLE), offset 0xD08

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

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

Vector Table Offset (VTABLE)

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

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

Bit/Field	Name	Type	Reset	Description
31:30	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
29	BASE	R/W	0	Vector Table Base
	Value	Description		
	0	The vector table is in the code memory region.		
	1	The vector table is in the SRAM memory region.		
28:9	OFFSET	R/W	0x000.00	Vector Table Offset When configuring the OFFSET field, the offset must be aligned to the number of exception entries in the vector table. Because there are 54 interrupts, the minimum alignment is 128 words.
8:0	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 33: Application Interrupt and Reset Control (APINT), offset 0xD0C

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

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

The PRIGROUP field indicates the position of the binary point that splits the INT_x fields in the **Interrupt Priority (PRIx)** registers into separate group priority and subpriority fields. Table 3-8 on page 151 shows how the PRIGROUP value controls this split. The bit numbers in the Group Priority Field and Subpriority Field columns in the table refer to the bits in the INTA field. For the INTB field, the corresponding bits are 15:13; for INTC, 23:21; and for INTD, 31:29.

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

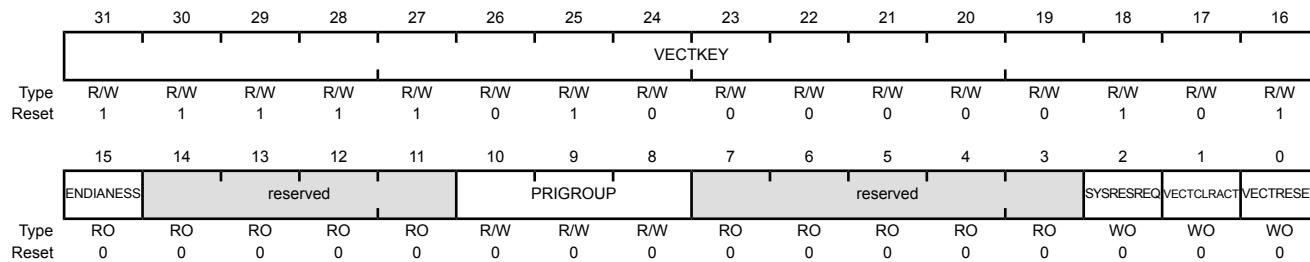
Table 3-8. Interrupt Priority Levels

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

a. INT_x field showing the binary point. An x denotes a group priority field bit, and a y denotes a subpriority field bit.

Application Interrupt and Reset Control (APINT)

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



Bit/Field	Name	Type	Reset	Description
31:16	VECTKEY	R/W	0xFA05	Register Key This field is used to guard against accidental writes to this register. 0x05FA must be written to this field in order to change the bits in this register. On a read, 0xFA05 is returned.
15	ENDIANESS	RO	0	Data Endianess The Stellaris implementation uses only little-endian mode so this is cleared to 0.
14:11	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description						
10:8	PRIGROUP	R/W	0x0	Interrupt Priority Grouping This field determines the split of group priority from subpriority (see Table 3-8 on page 151 for more information).						
7:3	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
2	SYSRESREQ	WO	0	System Reset Request <table border="0" style="margin-left: 20px;"> <tr> <th style="text-align: left;">Value</th> <th style="text-align: left;">Description</th> </tr> <tr> <td style="text-align: left;">0</td> <td>No effect.</td> </tr> <tr> <td style="text-align: left;">1</td> <td>Resets the core and all on-chip peripherals except the Debug interface.</td> </tr> </table> This bit is automatically cleared during the reset of the core and reads as 0.	Value	Description	0	No effect.	1	Resets the core and all on-chip peripherals except the Debug interface.
Value	Description									
0	No effect.									
1	Resets the core and all on-chip peripherals except the Debug interface.									
1	VECTCLRACT	WO	0	Clear Active NMI / Fault This bit is reserved for Debug use and reads as 0. This bit must be written as a 0, otherwise behavior is unpredictable.						
0	VECTRESET	WO	0	System Reset This bit is reserved for Debug use and reads as 0. This bit must be written as a 0, otherwise behavior is unpredictable.						

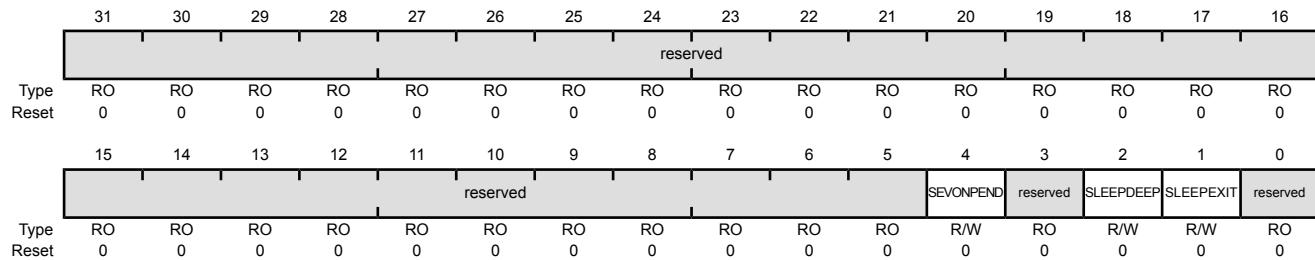
Register 34: System Control (SYSCTRL), offset 0xD10

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

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

System Control (SYSCTRL)

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



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

Bit/Field	Name	Type	Reset	Description				
1	SLEEP EXIT	R/W	0	<p>Sleep on ISR Exit</p> <p>Value Description</p> <table><tr><td>0</td><td>When returning from Handler mode to Thread mode, do not sleep when returning to Thread mode.</td></tr><tr><td>1</td><td>When returning from Handler mode to Thread mode, enter sleep or deep sleep on return from an ISR.</td></tr></table> <p>Setting this bit enables an interrupt-driven application to avoid returning to an empty main application.</p>	0	When returning from Handler mode to Thread mode, do not sleep when returning to Thread mode.	1	When returning from Handler mode to Thread mode, enter sleep or deep sleep on return from an ISR.
0	When returning from Handler mode to Thread mode, do not sleep when returning to Thread mode.							
1	When returning from Handler mode to Thread mode, enter sleep or deep sleep on return from an ISR.							
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.				

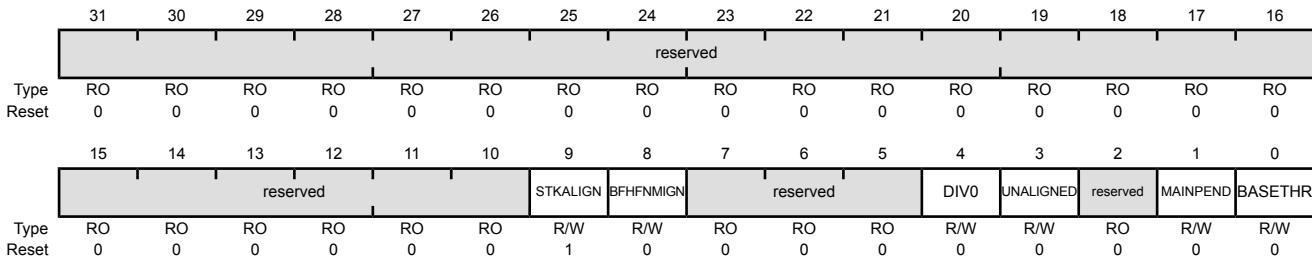
Register 35: Configuration and Control (CFGCTRL), offset 0xD14

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

The **CFGCTRL** register controls entry to Thread mode and enables: the handlers for NMI, hard fault and faults escalated by the **FAULTMASK** register to ignore bus faults; trapping of divide by zero and unaligned accesses; and access to the **SWTRIG** register by unprivileged software (see page 143).

Configuration and Control (CFGCTRL)

Base 0xE000.E000
Offset 0xD14
Type R/W, reset 0x0000.0200



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	STKALIGN	R/W	1	<p>Stack Alignment on Exception Entry</p> <p>Value Description</p> <p>0 The stack is 4-byte aligned.</p> <p>1 The stack is 8-byte aligned.</p> <p>On exception entry, the processor uses bit 9 of the stacked PSR to indicate the stack alignment. On return from the exception, it uses this stacked bit to restore the correct stack alignment.</p>
8	BFHFNMIGN	R/W	0	<p>Ignore Bus Fault in NMI and Fault</p> <p>This bit enables handlers with priority -1 or -2 to ignore data bus faults caused by load and store instructions. The setting of this bit applies to the hard fault, NMI, and FAULTMASK escalated handlers.</p> <p>Value Description</p> <p>0 Data bus faults caused by load and store instructions cause a lock-up.</p> <p>1 Handlers running at priority -1 and -2 ignore data bus faults caused by load and store instructions.</p> <p>Set this bit only when the handler and its data are in absolutely safe memory. The normal use of this bit is to probe system devices and bridges to detect control path problems and fix them.</p>
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description						
4	DIV0	R/W	0	<p>Trap on Divide by 0</p> <p>This bit enables faulting or halting when the processor executes an SDIV or UDIV instruction with a divisor of 0.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Do not trap on divide by 0. A divide by zero returns a quotient of 0.</td></tr> <tr> <td>1</td><td>Trap on divide by 0.</td></tr> </tbody> </table>	Value	Description	0	Do not trap on divide by 0. A divide by zero returns a quotient of 0.	1	Trap on divide by 0.
Value	Description									
0	Do not trap on divide by 0. A divide by zero returns a quotient of 0.									
1	Trap on divide by 0.									
3	UNALIGNED	R/W	0	<p>Trap on Unaligned Access</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Do not trap on unaligned halfword and word accesses.</td></tr> <tr> <td>1</td><td>Trap on unaligned halfword and word accesses. An unaligned access generates a usage fault.</td></tr> </tbody> </table> <p>Unaligned LDM, STM, LDRD, and STRD instructions always fault regardless of whether UNALIGNED is set.</p>	Value	Description	0	Do not trap on unaligned halfword and word accesses.	1	Trap on unaligned halfword and word accesses. An unaligned access generates a usage fault.
Value	Description									
0	Do not trap on unaligned halfword and word accesses.									
1	Trap on unaligned halfword and word accesses. An unaligned access generates a usage fault.									
2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
1	MAINPEND	R/W	0	<p>Allow Main Interrupt Trigger</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Disables unprivileged software access to the SWTRIG register.</td></tr> <tr> <td>1</td><td>Enables unprivileged software access to the SWTRIG register (see page 143).</td></tr> </tbody> </table>	Value	Description	0	Disables unprivileged software access to the SWTRIG register.	1	Enables unprivileged software access to the SWTRIG register (see page 143).
Value	Description									
0	Disables unprivileged software access to the SWTRIG register.									
1	Enables unprivileged software access to the SWTRIG register (see page 143).									
0	BASETHR	R/W	0	<p>Thread State Control</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The processor can enter Thread mode only when no exception is active.</td></tr> <tr> <td>1</td><td>The processor can enter Thread mode from any level under the control of an EXC_RETURN value (see “Exception Return” on page 108 for more information).</td></tr> </tbody> </table>	Value	Description	0	The processor can enter Thread mode only when no exception is active.	1	The processor can enter Thread mode from any level under the control of an EXC_RETURN value (see “Exception Return” on page 108 for more information).
Value	Description									
0	The processor can enter Thread mode only when no exception is active.									
1	The processor can enter Thread mode from any level under the control of an EXC_RETURN value (see “Exception Return” on page 108 for more information).									

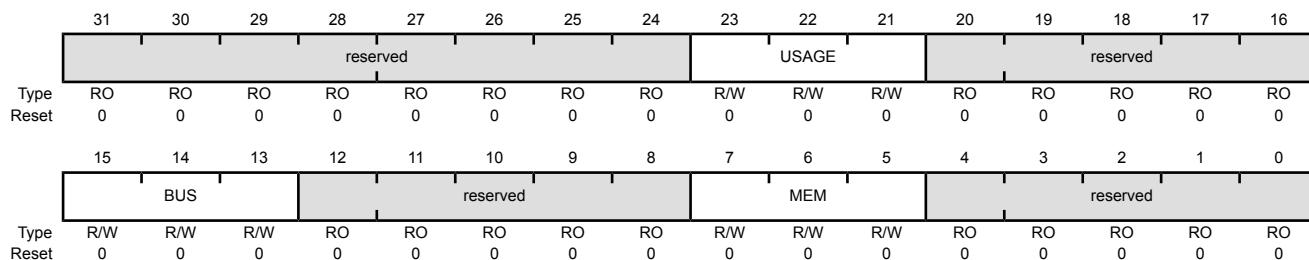
Register 36: System Handler Priority 1 (SYSPRI1), offset 0xD18

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

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

System Handler Priority 1 (SYSPRI1)

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



Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:21	USAGE	R/W	0x0	Usage Fault Priority This field configures the priority level of the usage fault. Configurable priority values are in the range 0-7, with lower values having higher priority.
20:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:13	BUS	R/W	0x0	Bus Fault Priority This field configures the priority level of the bus fault. Configurable priority values are in the range 0-7, with lower values having higher priority.
12:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	MEM	R/W	0x0	Memory Management Fault Priority This field configures the priority level of the memory management fault. Configurable priority values are in the range 0-7, with lower values having higher priority.
4:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

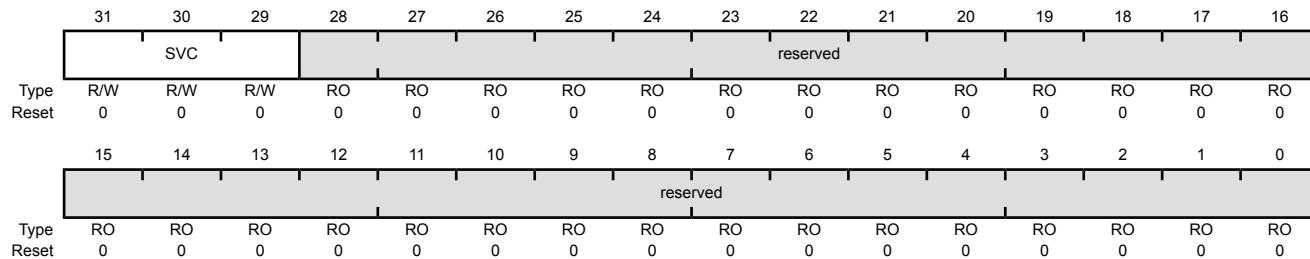
Register 37: System Handler Priority 2 (SYSPRI2), offset 0xD1C

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

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

System Handler Priority 2 (SYSPRI2)

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



Bit/Field	Name	Type	Reset	Description
31:29	SVC	R/W	0x0	SVCall Priority This field configures the priority level of SVCall. Configurable priority values are in the range 0-7, with lower values having higher priority.
28:0	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 38: System Handler Priority 3 (SYSPRI3), offset 0xD20

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

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

System Handler Priority 3 (SYSPRI3)

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

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

Bit/Field	Name	Type	Reset	Description
31:29	TICK	R/W	0x0	SysTick Exception Priority This field configures the priority level of the SysTick exception. Configurable priority values are in the range 0-7, with lower values having higher priority.
28:24	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:21	PENDSV	R/W	0x0	PendSV Priority This field configures the priority level of PendSV. Configurable priority values are in the range 0-7, with lower values having higher priority.
20:8	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	DEBUG	R/W	0x0	Debug Priority This field configures the priority level of Debug. Configurable priority values are in the range 0-7, with lower values having higher priority.
4:0	reserved	RO	0x0.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 39: System Handler Control and State (SYSHNDCTRL), offset 0xD24

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

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

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

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

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

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

System Handler Control and State (SYSHNDCTRL)

Base 0xE000.E000
Offset 0xD24
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	SVC	BUSP	MEMP	USAGEP	TICK	PNDSV	reserved	MON	SVCA	reserved	reserved	USGA	reserved	BUSA	MEMA	
Reset	R/W	R/W	R/W	R/W	R/W	R/W	RO	R/W	R/W	RO	RO	R/W	RO	R/W	R/W	R/W

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

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

Bit/Field	Name	Type	Reset	Description						
10	PNDSV	R/W	0	<p>PendSV Exception Active</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>A PendSV exception is not active.</td></tr> <tr> <td>1</td><td>A PendSV exception is active.</td></tr> </tbody> </table> <p>This bit can be modified to change the active status of the PendSV exception, however, see the Caution above before setting this bit.</p>	Value	Description	0	A PendSV exception is not active.	1	A PendSV exception is active.
Value	Description									
0	A PendSV exception is not active.									
1	A PendSV exception is active.									
9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
8	MON	R/W	0	<p>Debug Monitor Active</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The Debug monitor is not active.</td></tr> <tr> <td>1</td><td>The Debug monitor is active.</td></tr> </tbody> </table>	Value	Description	0	The Debug monitor is not active.	1	The Debug monitor is active.
Value	Description									
0	The Debug monitor is not active.									
1	The Debug monitor is active.									
7	SVCA	R/W	0	<p>SVC Call Active</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>SVC call is not active.</td></tr> <tr> <td>1</td><td>SVC call is active.</td></tr> </tbody> </table> <p>This bit can be modified to change the active status of the SVC call exception, however, see the Caution above before setting this bit.</p>	Value	Description	0	SVC call is not active.	1	SVC call is active.
Value	Description									
0	SVC call is not active.									
1	SVC call is active.									
6:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
3	USGA	R/W	0	<p>Usage Fault Active</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Usage fault is not active.</td></tr> <tr> <td>1</td><td>Usage fault is active.</td></tr> </tbody> </table> <p>This bit can be modified to change the active status of the usage fault exception, however, see the Caution above before setting this bit.</p>	Value	Description	0	Usage fault is not active.	1	Usage fault is active.
Value	Description									
0	Usage fault is not active.									
1	Usage fault is active.									
2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
1	BUSA	R/W	0	<p>Bus Fault Active</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Bus fault is not active.</td></tr> <tr> <td>1</td><td>Bus fault is active.</td></tr> </tbody> </table> <p>This bit can be modified to change the active status of the bus fault exception, however, see the Caution above before setting this bit.</p>	Value	Description	0	Bus fault is not active.	1	Bus fault is active.
Value	Description									
0	Bus fault is not active.									
1	Bus fault is active.									

Bit/Field	Name	Type	Reset	Description
0	MEMA	R/W	0	Memory Management Fault Active
<p style="text-align: center;">Value Description</p>				
<p style="text-align: center;">0 Memory management fault is not active. 1 Memory management fault is active.</p>				
<p style="text-align: center;">This bit can be modified to change the active status of the memory management fault exception, however, see the Caution above before setting this bit.</p>				

Register 40: Configurable Fault Status (FAULTSTAT), offset 0xD28

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

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

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

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

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

Bits are cleared by writing a 1 to them.

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

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

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

Configurable Fault Status (FAULTSTAT)

Base 0xE000.E000

Offset 0xD28

Type R/W1C, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description						
25	DIV0	R/W1C	0	<p>Divide-by-Zero Usage Fault</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No divide-by-zero fault has occurred, or divide-by-zero trapping is not enabled.</td></tr> <tr> <td>1</td><td>The processor has executed an SDIV or UDIV instruction with a divisor of 0.</td></tr> </tbody> </table> <p>When this bit is set, the PC value stacked for the exception return points to the instruction that performed the divide by zero. Trapping on divide-by-zero is enabled by setting the DIV0 bit in the Configuration and Control (CFGCTRL) register (see page 155). This bit is cleared by writing a 1 to it.</p>	Value	Description	0	No divide-by-zero fault has occurred, or divide-by-zero trapping is not enabled.	1	The processor has executed an SDIV or UDIV instruction with a divisor of 0.
Value	Description									
0	No divide-by-zero fault has occurred, or divide-by-zero trapping is not enabled.									
1	The processor has executed an SDIV or UDIV instruction with a divisor of 0.									
24	UNALIGN	R/W1C	0	<p>Unaligned Access Usage Fault</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No unaligned access fault has occurred, or unaligned access trapping is not enabled.</td></tr> <tr> <td>1</td><td>The processor has made an unaligned memory access.</td></tr> </tbody> </table> <p>Unaligned LDM, STM, LDRD, and STRD instructions always fault regardless of the configuration of this bit. Trapping on unaligned access is enabled by setting the UNALIGNED bit in the CFGCTRL register (see page 155). This bit is cleared by writing a 1 to it.</p>	Value	Description	0	No unaligned access fault has occurred, or unaligned access trapping is not enabled.	1	The processor has made an unaligned memory access.
Value	Description									
0	No unaligned access fault has occurred, or unaligned access trapping is not enabled.									
1	The processor has made an unaligned memory access.									
23:20	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
19	NOCP	R/W1C	0	<p>No Coprocessor Usage Fault</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>A usage fault has not been caused by attempting to access a coprocessor.</td></tr> <tr> <td>1</td><td>The processor has attempted to access a coprocessor.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to it.</p>	Value	Description	0	A usage fault has not been caused by attempting to access a coprocessor.	1	The processor has attempted to access a coprocessor.
Value	Description									
0	A usage fault has not been caused by attempting to access a coprocessor.									
1	The processor has attempted to access a coprocessor.									
18	INVPC	R/W1C	0	<p>Invalid PC Load Usage Fault</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>A usage fault has not been caused by attempting to load an invalid PC value.</td></tr> <tr> <td>1</td><td>The processor has attempted an illegal load of EXC_RETURN to the PC as a result of an invalid context or an invalid EXC_RETURN value.</td></tr> </tbody> </table> <p>When this bit is set, the PC value stacked for the exception return points to the instruction that tried to perform the illegal load of the PC. This bit is cleared by writing a 1 to it.</p>	Value	Description	0	A usage fault has not been caused by attempting to load an invalid PC value.	1	The processor has attempted an illegal load of EXC_RETURN to the PC as a result of an invalid context or an invalid EXC_RETURN value.
Value	Description									
0	A usage fault has not been caused by attempting to load an invalid PC value.									
1	The processor has attempted an illegal load of EXC_RETURN to the PC as a result of an invalid context or an invalid EXC_RETURN value.									

Bit/Field	Name	Type	Reset	Description
17	INVSTAT	R/W1C	0	<p>Invalid State Usage Fault</p> <p>Value Description</p> <p>0 A usage fault has not been caused by an invalid state.</p> <p>1 The processor has attempted to execute an instruction that makes illegal use of the EPSR register.</p> <p>When this bit is set, the PC value stacked for the exception return points to the instruction that attempted the illegal use of the Execution Program Status Register (EPSR) register.</p> <p>This bit is not set if an undefined instruction uses the EPSR register.</p> <p>This bit is cleared by writing a 1 to it.</p>
16	UNDEF	R/W1C	0	<p>Undefined Instruction Usage Fault</p> <p>Value Description</p> <p>0 A usage fault has not been caused by an undefined instruction.</p> <p>1 The processor has attempted to execute an undefined instruction.</p> <p>When this bit is set, the PC value stacked for the exception return points to the undefined instruction.</p> <p>An undefined instruction is an instruction that the processor cannot decode.</p> <p>This bit is cleared by writing a 1 to it.</p>
15	BFARV	R/W1C	0	<p>Bus Fault Address Register Valid</p> <p>Value Description</p> <p>0 The value in the Bus Fault Address (FAULTADDR) register is not a valid fault address.</p> <p>1 The FAULTADDR register is holding a valid fault address.</p> <p>This bit is set after a bus fault, where the address is known. Other faults can clear this bit, such as a memory management fault occurring later. If a bus fault occurs and is escalated to a hard fault because of priority, the hard fault handler must clear this bit. This action prevents problems if returning to a stacked active bus fault handler whose FAULTADDR register value has been overwritten.</p> <p>This bit is cleared by writing a 1 to it.</p>
14:13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	BSTKE	R/W1C	0	<p>Stack Bus Fault</p> <p>Value Description</p> <p>0 No bus fault has occurred on stacking for exception entry.</p> <p>1 Stacking for an exception entry has caused one or more bus faults.</p> <p>When this bit is set, the SP is still adjusted but the values in the context area on the stack might be incorrect. A fault address is not written to the FAULTADDR register.</p> <p>This bit is cleared by writing a 1 to it.</p>

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

Bit/Field	Name	Type	Reset	Description
7	MMARV	R/W1C	0	<p>Memory Management Fault Address Register Valid</p> <p>Value Description</p> <p>0 The value in the Memory Management Fault Address (MMADDR) register is not a valid fault address.</p> <p>1 The MMADDR register is holding a valid fault address.</p> <p>If a memory management fault occurs and is escalated to a hard fault because of priority, the hard fault handler must clear this bit. This action prevents problems if returning to a stacked active memory management fault handler whose MMADDR register value has been overwritten. This bit is cleared by writing a 1 to it.</p>
6:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	MSTKE	R/W1C	0	<p>Stack Access Violation</p> <p>Value Description</p> <p>0 No memory management fault has occurred on stacking for exception entry.</p> <p>1 Stacking for an exception entry has caused one or more access violations.</p> <p>When this bit is set, the SP is still adjusted but the values in the context area on the stack might be incorrect. A fault address is not written to the MMADDR register.</p> <p>This bit is cleared by writing a 1 to it.</p>
3	MUSTKE	R/W1C	0	<p>Unstack Access Violation</p> <p>Value Description</p> <p>0 No memory management fault has occurred on unstacking for a return from exception.</p> <p>1 Unstacking for a return from exception has caused one or more access violations.</p> <p>This fault is chained to the handler. Thus, when this bit is set, the original return stack is still present. The SP is not adjusted from the failing return, a new save is not performed, and a fault address is not written to the MMADDR register.</p> <p>This bit is cleared by writing a 1 to it.</p>
2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description						
1	DERR	R/W1C	0	<p>Data Access Violation</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>A data access violation has not occurred.</td></tr> <tr> <td>1</td><td>The processor attempted a load or store at a location that does not permit the operation.</td></tr> </tbody> </table> <p>When this bit is set, the PC value stacked for the exception return points to the faulting instruction and the address of the attempted access is written to the MMADDR register. This bit is cleared by writing a 1 to it.</p>	Value	Description	0	A data access violation has not occurred.	1	The processor attempted a load or store at a location that does not permit the operation.
Value	Description									
0	A data access violation has not occurred.									
1	The processor attempted a load or store at a location that does not permit the operation.									
0	IERR	R/W1C	0	<p>Instruction Access Violation</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>An instruction access violation has not occurred.</td></tr> <tr> <td>1</td><td>The processor attempted an instruction fetch from a location that does not permit execution.</td></tr> </tbody> </table> <p>This fault occurs on any access to an XN region, even when the MPU is disabled or not present. When this bit is set, the PC value stacked for the exception return points to the faulting instruction and the address of the attempted access is not written to the MMADDR register. This bit is cleared by writing a 1 to it.</p>	Value	Description	0	An instruction access violation has not occurred.	1	The processor attempted an instruction fetch from a location that does not permit execution.
Value	Description									
0	An instruction access violation has not occurred.									
1	The processor attempted an instruction fetch from a location that does not permit execution.									

Register 41: Hard Fault Status (HFAULTSTAT), offset 0xD2C

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

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

Bits are cleared by writing a 1 to them.

Hard Fault Status (HFAULTSTAT)

Base 0xE000.E000
Offset 0xD2C
Type R/W1C, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31	DBG	R/W1C	0	Debug Event This bit is reserved for Debug use. This bit must be written as a 0, otherwise behavior is unpredictable.
30	FORCED	R/W1C	0	Forced Hard Fault Value Description 0 No forced hard fault has occurred. 1 A forced hard fault has been generated by escalation of a fault with configurable priority that cannot be handled, either because of priority or because it is disabled. When this bit is set, the hard fault handler must read the other fault status registers to find the cause of the fault. This bit is cleared by writing a 1 to it.
29:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	VECT	R/W1C	0	Vector Table Read Fault Value Description 0 No bus fault has occurred on a vector table read. 1 A bus fault occurred on a vector table read. This error is always handled by the hard fault handler. When this bit is set, the PC value stacked for the exception return points to the instruction that was preempted by the exception. This bit is cleared by writing a 1 to it.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 42: Memory Management Fault Address (MMADDR), offset 0xD34

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

The **MMADDR** register contains the address of the location that generated a memory management fault. When an unaligned access faults, the address in the **MMADDR** register is the actual address that faulted. Because a single read or write instruction can be split into multiple aligned accesses, the fault address can be any address in the range of the requested access size. Bits in the **Memory Management Fault Status (MFAULTSTAT)** register indicate the cause of the fault and whether the value in the **MMADDR** register is valid (see page 164).

Memory Management Fault Address (MMADDR)

Base 0xE000.E000

Offset 0xD34

Type R/W, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR																
Type	R/W															
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ADDR																
Type	R/W															
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

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

Register 43: Bus Fault Address (FAULTADDR), offset 0xD38

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

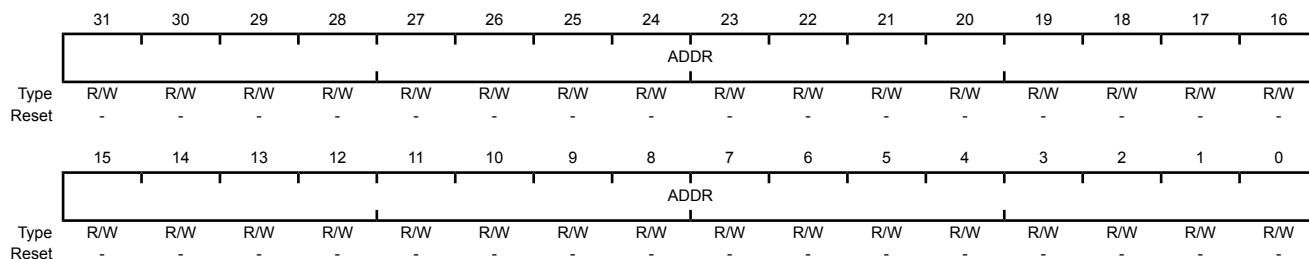
The **FAULTADDR** register contains the address of the location that generated a bus fault. When an unaligned access faults, the address in the **FAULTADDR** register is the one requested by the instruction, even if it is not the address of the fault. Bits in the **Bus Fault Status (BFAULTSTAT)** register indicate the cause of the fault and whether the value in the **FAULTADDR** register is valid (see page 164).

Bus Fault Address (FAULTADDR)

Base 0xE000.E000

Offset 0xD38

Type R/W, reset -



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

3.6 Memory Protection Unit (MPU) Register Descriptions

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

The MPU registers can only be accessed from privileged mode.

Register 44: MPU Type (MPUTYPE), offset 0xD90

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

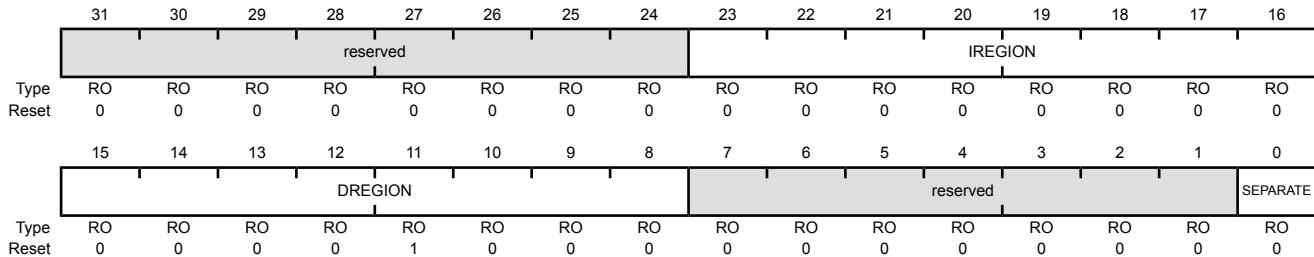
The **MPUTYPE** register indicates whether the MPU is present, and if so, how many regions it supports.

MPU Type (MPUTYPE)

Base 0xE000.E000

Offset 0xD90

Type RO, reset 0x0000.0800



Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:16	IREGION	RO	0x00	Number of I Regions This field indicates the number of supported MPU instruction regions. This field always contains 0x00. The MPU memory map is unified and is described by the DREGION field.
15:8	DREGION	RO	0x08	Number of D Regions Value Description 0x08 Indicates there are eight supported MPU data regions.
7:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	SEPARATE	RO	0	Separate or Unified MPU Value Description 0 Indicates the MPU is unified.

Register 45: MPU Control (MPUCTRL), offset 0xD94

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

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

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

- For privileged accesses, the default memory map is as described in “Memory Model” on page 92. Any access by privileged software that does not address an enabled memory region behaves as defined by the default memory map.
 - Any access by unprivileged software that does not address an enabled memory region causes a memory management fault.

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

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

When the `ENABLE` bit is clear, the system uses the default memory map, which has the same memory attributes as if the MPU is not implemented (see Table 2-5 on page 95 for more information). The default memory map applies to accesses from both privileged and unprivileged software.

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

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

MPU Control (MPUCTRL)

Base 0xE000.E000

Offset 0xD94

Type R/W, reset 0x0000.0000

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

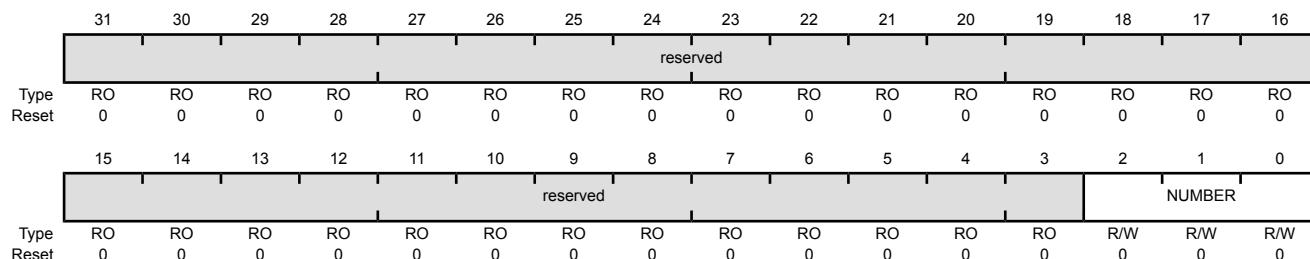
Register 46: MPU Region Number (MPUNUMBER), offset 0xD98

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

The **MPUNUMBER** register selects which memory region is referenced by the **MPU Region Base Address (MPUBASE)** and **MPU Region Attribute and Size (MPUATTR)** registers. Normally, the required region number should be written to this register before accessing the **MPUBASE** or the **MPUATTR** register. However, the region number can be changed by writing to the **MPUBASE** register with the **VALID** bit set (see page 177). This write updates the value of the **REGION** field.

MPU Region Number (MPUNUMBER)

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



Register 47: MPU Region Base Address (MPUBASE), offset 0xD9C**Register 48: MPU Region Base Address Alias 1 (MPUBASE1), offset 0xDA4****Register 49: MPU Region Base Address Alias 2 (MPUBASE2), offset 0xDAC****Register 50: MPU Region Base Address Alias 3 (MPUBASE3), offset 0xDB4**

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

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

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

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

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

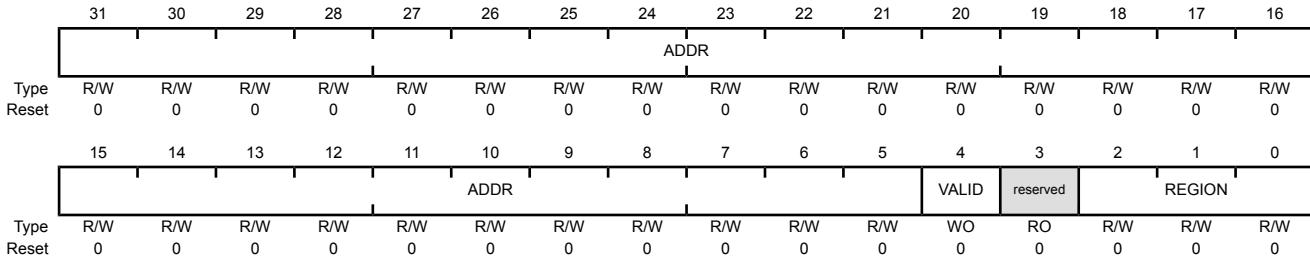
The base address is aligned to the size of the region. For example, a 64-KB region must be aligned on a multiple of 64 KB, for example, at 0x0001.0000 or 0x0002.0000.

MPU Region Base Address (MPUBASE)

Base 0xE000.E000

Offset 0xD9C

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:5	ADDR	R/W	0x0000.0000	<p>Base Address Mask</p> <p>Bits 31:N in this field contain the region base address. The value of N depends on the region size, as shown above. The remaining bits (N-1):5 are reserved.</p> <p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>

Bit/Field	Name	Type	Reset	Description						
4	VALID	WO	0	<p>Region Number Valid</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The MPUNUMBER register is not changed and the processor updates the base address for the region specified in the MPUNUMBER register and ignores the value of the REGION field.</td></tr> <tr> <td>1</td><td>The MPUNUMBER register is updated with the value of the REGION field and the base address is updated for the region specified in the REGION field.</td></tr> </tbody> </table> <p>This bit is always read as 0.</p>	Value	Description	0	The MPUNUMBER register is not changed and the processor updates the base address for the region specified in the MPUNUMBER register and ignores the value of the REGION field.	1	The MPUNUMBER register is updated with the value of the REGION field and the base address is updated for the region specified in the REGION field.
Value	Description									
0	The MPUNUMBER register is not changed and the processor updates the base address for the region specified in the MPUNUMBER register and ignores the value of the REGION field.									
1	The MPUNUMBER register is updated with the value of the REGION field and the base address is updated for the region specified in the REGION field.									
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
2:0	REGION	R/W	0x0	<p>Region Number</p> <p>On a write, contains the value to be written to the MPUNUMBER register.</p> <p>On a read, returns the current region number in the MPUNUMBER register.</p>						

Register 51: MPU Region Attribute and Size (MPUATTR), offset 0xDA0**Register 52: MPU Region Attribute and Size Alias 1 (MPUATTR1), offset 0xDA8****Register 53: MPU Region Attribute and Size Alias 2 (MPUATTR2), offset 0xDB0****Register 54: MPU Region Attribute and Size Alias 3 (MPUATTR3), offset 0xDB8**

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

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

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

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

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

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

The smallest permitted region size is 32 bytes, corresponding to a SIZE value of 4. Table 3-9 on page 179 gives example SIZE values with the corresponding region size and value of N in the **MPU Region Base Address (MPUBASE)** register.

Table 3-9. Example SIZE Field Values

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

a. Refers to the N parameter in the **MPUBASE** register (see page 177).

MPU Region Attribute and Size (MPUATTR)

Base 0xE000.E000

Offset 0xDA0

Type R/W, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved		XN	reserved		AP		reserved		TEX		S	C	B	
Type	RO	RO	RO	R/W	RO	R/W	R/W	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				SRD				reserved		SIZE			ENABLE		
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	XN	R/W	0	Instruction Access Disable Value Description 0 Instruction fetches are enabled. 1 Instruction fetches are disabled.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26:24	AP	R/W	0	Access Privilege For information on using this bit field, see Table 3-5 on page 123.
23:22	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
21:19	TEX	R/W	0x0	Type Extension Mask For information on using this bit field, see Table 3-3 on page 122.
18	S	R/W	0	Shareable For information on using this bit, see Table 3-3 on page 122.
17	C	R/W	0	Cacheable For information on using this bit, see Table 3-3 on page 122.
16	B	R/W	0	Bufferable For information on using this bit, see Table 3-3 on page 122.
15:8	SRD	R/W	0x00	Subregion Disable Bits Value Description 0 The corresponding subregion is enabled. 1 The corresponding subregion is disabled. Region sizes of 128 bytes and less do not support subregions. When writing the attributes for such a region, configure the SRD field as 0x00. See the section called "Subregions" on page 121 for more information.
7:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:1	SIZE	R/W	0x0	Region Size Mask The SIZE field defines the size of the MPU memory region specified by the MPUNUMBER register. Refer to Table 3-9 on page 179 for more information.

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

4 JTAG Interface

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

The JTAG port is comprised of 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 Stellaris® 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 Stellaris JTAG instructions select the Stellaris TDO output. The multiplexer is controlled by the Stellaris JTAG controller, which has comprehensive programming for the ARM, Stellaris, and unimplemented JTAG instructions.

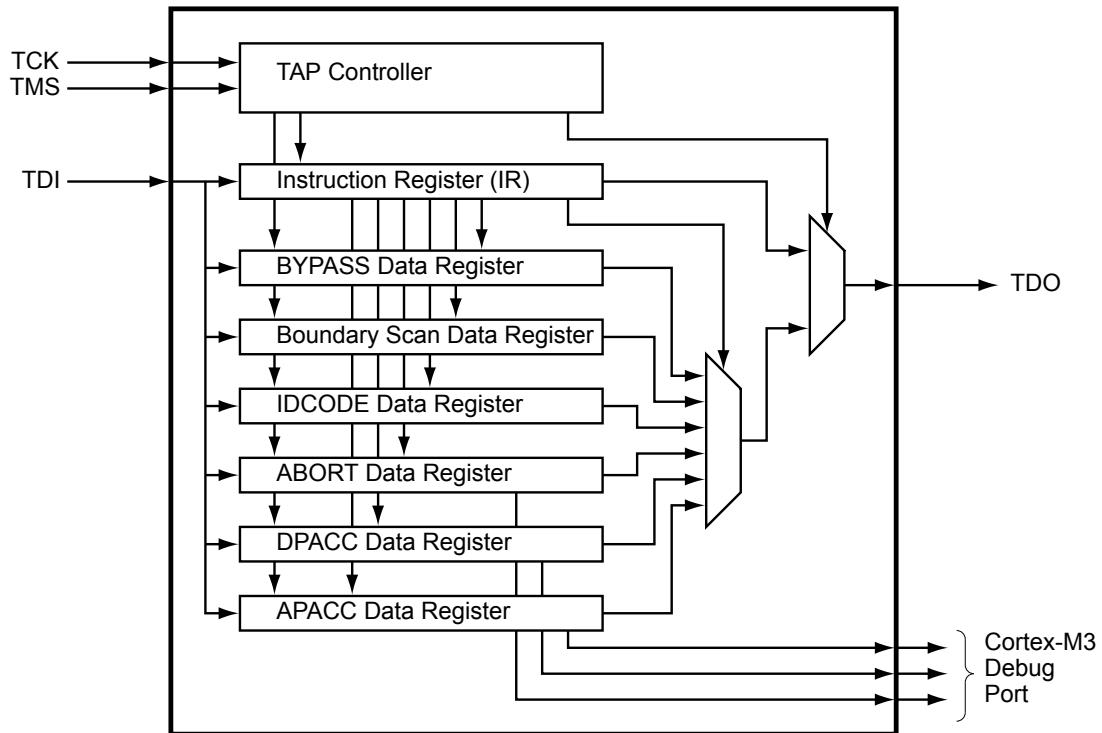
The Stellaris JTAG module has the following features:

- IEEE 1149.1-1990 compatible Test Access Port (TAP) controller
- Four-bit Instruction Register (IR) chain for storing JTAG instructions
- IEEE standard instructions: BYPASS, IDCODE, SAMPLE/PRELOAD, EXTEST and INTEST
- ARM additional instructions: APACC, DPACC and ABORT
- Integrated ARM Serial Wire Debug (SWD)
 - 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® Debug Interface V5 Architecture Specification* for more information on the ARM JTAG controller.

4.1 Block Diagram

Figure 4-1. JTAG Module Block Diagram



4.2 Signal Description

Table 4-1 on page 183 and Table 4-2 on page 184 list the external signals of the JTAG/SWD controller and describe the function of each. The JTAG/SWD controller signals are alternate functions for some GPIO signals, however note that the reset state of the pins is for the JTAG/SWD function. The JTAG/SWD controller signals are under commit protection and require a special process to be configured as GPIOs, see “Commit Control” on page 431. The column in the table below titled “Pin Mux/Pin Assignment” lists the GPIO pin placement for the JTAG/SWD controller signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 447) is set to choose the JTAG/SWD function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 465) to assign the JTAG/SWD controller signals to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 423.

Table 4-1. Signals for JTAG_SWD_SWO (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
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.

Table 4-1. Signals for JTAG_SWD_SWO (100LQFP) (continued)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
TMS	79	PC1 (3)	I	TTL	JTAG TMS and SWDIO.

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

Table 4-2. Signals for JTAG_SWD_SWO (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
SWCLK	A9	PC0 (3)	I	TTL	JTAG/SWD CLK.
SWDIO	B9	PC1 (3)	I/O	TTL	JTAG TMS and SWDIO.
SWO	A10	PC3 (3)	O	TTL	JTAG TDO and SWO.
TCK	A9	PC0 (3)	I	TTL	JTAG/SWD CLK.
TDI	B8	PC2 (3)	I	TTL	JTAG TDI.
TDO	A10	PC3 (3)	O	TTL	JTAG TDO and SWO.
TMS	B9	PC1 (3)	I	TTL	JTAG TMS and SWDIO.

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

4.3 Functional Description

A high-level conceptual drawing of the JTAG module is shown in Figure 4-1 on page 183. 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 4-4 on page 190 for a list of implemented instructions).

See “JTAG and Boundary Scan” on page 1238 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{RST} input and POR, whereas the internal JTAG logic is only reset with POR. See “Reset Sources” on page 195 for more information on reset.

4.3.1 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{RST} input are given in Table 4-3. Detailed information on each pin follows. Refer to “General-Purpose Input/Outputs (GPIOs)” on page 423 for information on how to reprogram the configuration of these pins.

Table 4-3. JTAG Port Pins State after Power-On Reset or \overline{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

4.3.1.1 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 453 and page 455).

4.3.1.2 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 4-2 on page 186.

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 453).

4.3.1.3 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 453).

4.3.1.4 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

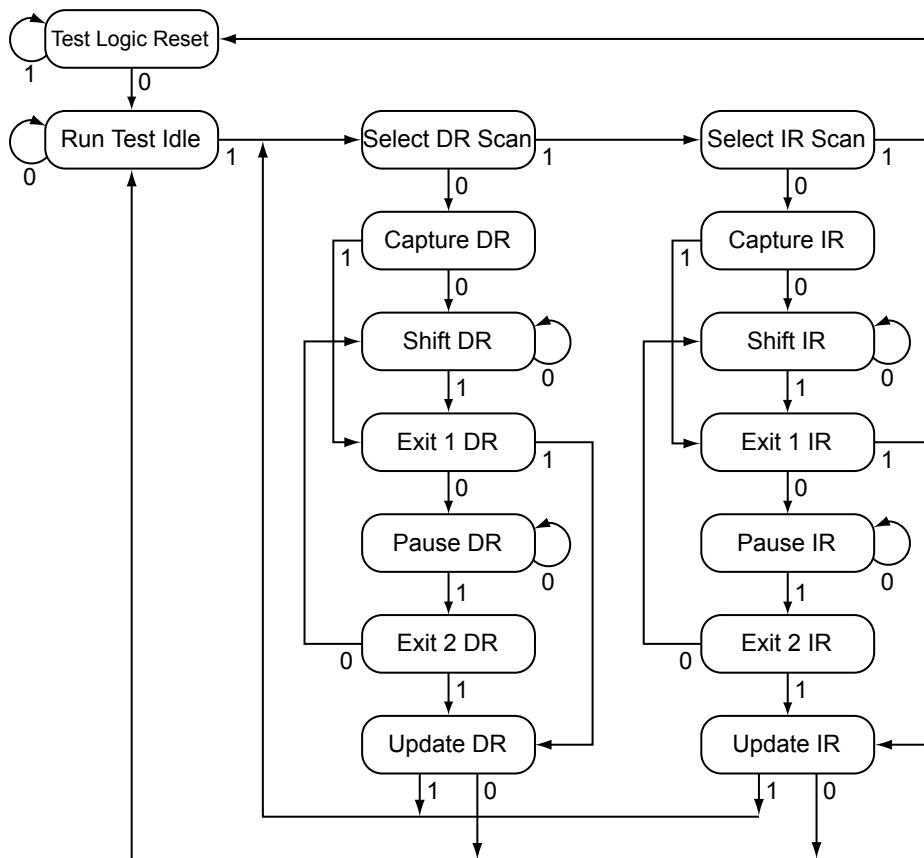
chain being accessed. In order to save power when the JTAG port is not being used, the TDO pin is placed in an inactive drive state when not actively shifting out data. Because TDO can be connected to the TDI of another controller in a daisy-chain configuration, the *IEEE Standard 1149.1* expects the 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 453 and page 455).

4.3.2 JTAG TAP Controller

The JTAG TAP controller state machine is shown in Figure 4-2. 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 4-2. Test Access Port State Machine



4.3.3 Shift Registers

The Shift Registers consist of a serial shift register chain and a parallel load register. The serial shift register chain samples specific information during the TAP controller's CAPTURE states and allows

this information to be shifted out 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 190.

4.3.4 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.

4.3.4.1 GPIO Functionality

When the microcontroller is reset with either a POR or \overline{RST} , the JTAG/SWD port pins default to their JTAG/SWD configurations. The default configuration includes enabling digital functionality ($DEN[3:0]$ set in the **Port C GPIO Digital Enable (GPIODEN)** register), enabling the pull-up resistors ($PUE[3:0]$ set in the **Port C GPIO Pull-Up Select (GPIOPUR)** register), disabling the pull-down resistors ($PDE[3:0]$ cleared in the **Port C GPIO Pull-Down Select (GPIOPDR)** register) and enabling the alternate hardware function ($AFSEL[3:0]$ set in the **Port C GPIO Alternate Function Select (GPIOAFSEL)** register) on the JTAG/SWD pins. See page 447, page 453, page 455, and page 458.

It is possible for software to configure these pins as GPIOs after reset by clearing $AFSEL[3:0]$ 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 provided for the **NMI** pin ($PB7$) and the four JTAG/SWD pins ($PC[3:0]$). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 447), **GPIO Pull Up Select (GPIOPUR)** register (see page 453), **GPIO Pull-Down Select (GPIOPDR)** register (see page 455), and **GPIO Digital Enable (GPIODEN)** register (see page 458) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 460) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 461) have been set.

4.3.4.2 Communication with JTAG/SWD

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

4.3.4.3 Recovering a "Locked" Microcontroller

Note: Performing the sequence below restores the nonvolatile registers discussed in “Nonvolatile Register Programming” on page 335 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 port unlock 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 debug port unlock sequence 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 189.
3. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence on the section called “SWD-to-JTAG Switching” on page 189.
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.

4.3.4.4 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® Debug Interface V5 Architecture Specification*.

Because this sequence is a valid series of JTAG operations that could be issued, the ARM JTAG TAP controller is not fully compliant to the *IEEE Standard 1149.1*. This 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.

4.4 Initialization and Configuration

After a Power-On-Reset or an external reset (\overline{RST}), the JTAG pins are automatically configured for JTAG communication. No user-defined initialization or configuration is needed. However, if the user application changes these pins to their GPIO function, they must be configured back to their JTAG functionality before JTAG communication can be restored. 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.

4.5 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.

4.5.1 Instruction Register (IR)

The JTAG TAP Instruction Register (IR) is a four-bit serial scan chain connected between the JTAG TDI and TDO pins with a parallel load register. When the TAP Controller is placed in the correct states, bits can be shifted into the 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 4-4. A detailed explanation of each instruction, along with its associated Data Register, follows.

Table 4-4. JTAG Instruction Register Commands

IR[3:0]	Instruction	Description
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.
All Others	Reserved	Defaults to the BYPASS instruction to ensure that TDI is always connected to TDO.

4.5.1.1 EXTEST Instruction

The EXTEST instruction is not associated with its own Data Register chain. 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.

4.5.1.2 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.

4.5.1.3 SAMPLE/PRELOAD Instruction

The SAMPLE/PRELOAD instruction connects the Boundary Scan Data Register chain between TDI and TDO. This instruction samples the current state of the pad pins for observation and preloads new test data. Each GPIO pad has an associated input, output, and output enable signal. When the TAP controller enters the Capture DR state during this instruction, the input, output, and output-enable signals to each of the GPIO pads are captured. These samples are serially shifted out 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 192 for more information.

4.5.1.4 ABORT Instruction

The ABORT instruction connects the associated ABORT Data Register chain between TDI and TDO. This instruction provides read and write access to the ABORT Register of the ARM Debug Access Port (DAP). Shifting the proper data into this Data Register clears various error bits or initiates a DAP abort of a previous request. See the “ABORT Data Register” on page 193 for more information.

4.5.1.5 DPACC Instruction

The DPACC instruction connects the associated DPACC Data Register chain between TDI and TDO. This instruction provides read and write access to the DPACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to the ARM debug and status registers. See “DPACC Data Register” on page 193 for more information.

4.5.1.6 APACC Instruction

The APACC instruction connects the associated APACC Data Register chain between TDI and TDO. This instruction provides read and write access to the APACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to internal components and buses through the Debug Port. See “APACC Data Register” on page 193 for more information.

4.5.1.7 IDCODE Instruction

The IDCODE instruction connects the associated IDCODE Data Register chain between TDI and TDO. This instruction provides information on the manufacturer, part number, and version of the ARM core. This information can be used by testing equipment and debuggers to automatically configure 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 192 for more information.

4.5.1.8 BYPASS Instruction

The BYPASS instruction connects the associated BYPASS Data Register chain between TDI and TDO. This instruction is used to create a minimum length serial path between the TDI and TDO ports. The BYPASS Data Register is a single-bit shift register. This instruction improves test efficiency by allowing components that are not needed for a specific test to be bypassed in the JTAG scan chain by loading them with the BYPASS instruction. See “BYPASS Data Register” on page 192 for more information.

4.5.2 Data Registers

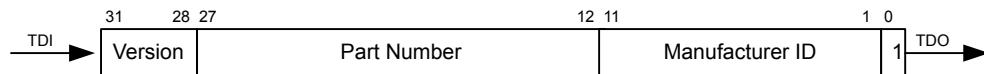
The JTAG module contains six Data Registers. These serial Data Register chains include: IDCODE, BYPASS, Boundary Scan, APACC, DPACC, and ABORT and are discussed in the following sections.

4.5.2.1 IDCODE Data Register

The format for the 32-bit IDCODE Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 4-3. 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 0x4BA0.0477. This value allows the debuggers to automatically configure themselves to work correctly with the Cortex-M3 during debug.

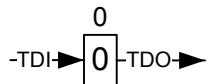
Figure 4-3. IDCODE Register Format



4.5.2.2 BYPASS Data Register

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



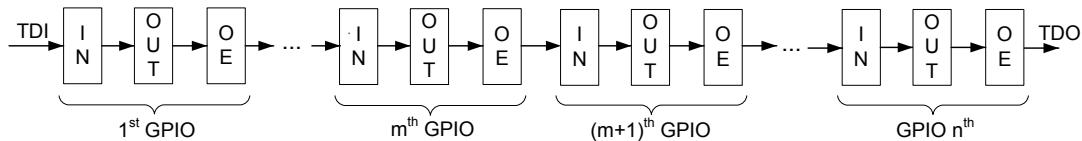
4.5.2.3 Boundary Scan Data Register

The format of the Boundary Scan Data Register is shown in Figure 4-5. 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.

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 4-5. Boundary Scan Register Format



4.5.2.4 APACC Data Register

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

4.5.2.5 DPACC Data Register

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

4.5.2.6 ABORT Data Register

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

5 System Control

System control 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.

5.1 Signal Description

Table 5-1 on page 194 and Table 5-2 on page 194 list the external signals of the System Control module and describe the function of each. The **NMI** signal is the alternate function for the GPIO PB7 signal and functions as a GPIO after reset. PB7 is under commit protection and requires a special process to be configured as the **NMI** signal or to subsequently return to the GPIO function, see “Commit Control” on page 431. The column in the table below titled “Pin Mux/Pin Assignment” lists the GPIO pin placement for the **NMI** signal. The **AFSEL** bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 447) should be set to choose the **NMI** function. The number in parentheses is the encoding that must be programmed into the **PMCn** field in the **GPIO Port Control (GPIOPCTL)** register (page 465) to assign the **NMI** signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 423. The remaining signals (with the word “fixed” in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

Table 5-1. Signals for System Control & Clocks (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
NMI	89	PB7 (4)	I	TTL	Non-maskable interrupt.
OSC0	48	fixed	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	49	fixed	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
RST	64	fixed	I	TTL	System reset input.

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

Table 5-2. Signals for System Control & Clocks (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
NMI	A8	PB7 (4)	I	TTL	Non-maskable interrupt.
OSC0	L11	fixed	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	M11	fixed	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
RST	H11	fixed	I	TTL	System reset input.

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

5.2 Functional Description

The System Control module provides the following capabilities:

- Device identification, see “Device Identification” on page 195

- Local control, such as reset (see “Reset Control” on page 195), power (see “Power Control” on page 203) and clock control (see “Clock Control” on page 203)
- System control (Run, Sleep, and Deep-Sleep modes), see “System Control” on page 211

5.2.1 Device Identification

Several read-only registers provide software with information on the microcontroller, such as version, part number, SRAM size, Flash memory size, and other features. See the **DID0** (page 215), **DID1** (page 245), **DC0-DC9** (page 247) and **NVMSTAT** (page 267) registers.

5.2.2 Reset Control

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

5.2.2.1 Reset Sources

The LM3S9L97 microcontroller has six sources of reset:

1. Power-on reset (POR) (see page 196).
2. External reset input pin (\overline{RST}) assertion (see page 196).
3. Internal brown-out (BOR) detector (see page 198).
4. Software-initiated reset (with the software reset registers) (see page 198).
5. A watchdog timer reset condition violation (see page 199).
6. MOSC failure (see page 202).

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

Table 5-3. Reset Sources

Reset Source	Core Reset?	JTAG Reset?	On-Chip Peripherals Reset? ^a
Power-On Reset	Yes	Yes	Yes
\overline{RST}	Yes	Yes	Yes
Brown-Out Reset	Yes	Yes	Yes
Software System Request Reset ^b	Yes	Yes	Yes
Software Peripheral Reset	No	Yes	Yes ^c
Watchdog Reset	Yes	Yes	Yes
MOSC Failure Reset	Yes	Yes	Yes

a. Refer to “Register Reset” on page 310 for information on how reset affects the Hibernation module.

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

c. 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.

At any reset that resets the core, the user has the opportunity to direct the core to execute the ROM Boot Loader or the application in Flash memory by using any GPIO signal as configured in the **Boot Configuration (BOOTCFG)** register.

At reset, the ROM is mapped over the Flash memory so that the ROM boot sequence is always executed. The boot sequence executed from ROM is as follows:

1. The **BA** bit (below) is cleared such that ROM is mapped to 0x01xx.xxxx and Flash memory is mapped to address 0x0.
2. The **BOOTCFG** register is read. If the **EN** bit is set, the status of the specified GPIO pin is compared with the specified polarity. If the status matches the specified polarity, the ROM Boot Loader is executed.
3. If the status doesn't match the specified polarity, the address at address 0x0000.0004 is checked to see if the Flash memory is programmed. If the data at this address is 0xFFFF.FFFF, the ROM Boot Loader is executed.
4. If there is valid data at address 0x0000.0004, the stack pointer (**SP**) is loaded from Flash memory at address 0x0000.0000 and the program counter (**PC**) is loaded from address 0x0000.0004. The user application begins executing.

For example, if the **BOOTCFG** register is written and committed with the value of 0x0000.3C01, then **PB7** is examined at reset to determine if the ROM Boot Loader should be executed. If **PB7** is Low, the core unconditionally begins executing the ROM boot loader. If **PB7** is High, then the application in Flash memory is executed if the reset vector at location 0x0000.0004 is not 0xFFFF.FFFF. Otherwise, the ROM boot loader is executed.

5.2.2.2 Power-On Reset (POR)

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

The internal Power-On Reset (POR) circuit monitors the power supply voltage (V_{DD}) and generates a reset signal to all of the internal logic including JTAG when the power supply ramp reaches a threshold value (V_{TH}). The microcontroller must be operating within the specified operating parameters when the on-chip power-on reset pulse is complete. For applications that require the use of an external reset signal to hold the microcontroller in reset longer than the internal POR, the \overline{RST} input may be used as discussed in “External \overline{RST} Pin” on page 196.

The Power-On Reset sequence is as follows:

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

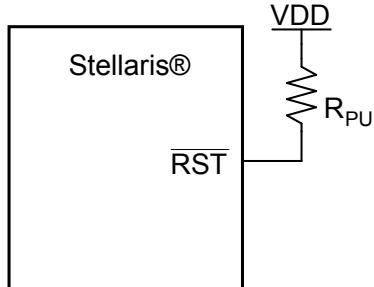
The internal POR is only active on the initial power-up of the microcontroller. The Power-On Reset timing is shown in Figure 26-6 on page 1240.

5.2.2.3 External \overline{RST} Pin

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

If the application only uses the internal POR circuit, the $\overline{\text{RST}}$ input must be connected to the power supply (V_{DD}) through an optional pull-up resistor (0 to 100K Ω) as shown in Figure 5-1 on page 197.

Figure 5-1. Basic $\overline{\text{RST}}$ Configuration



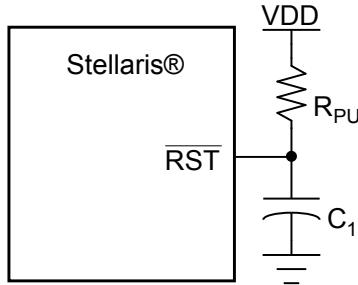
$$R_{PU} = 0 \text{ to } 100 \text{ k}\Omega$$

The external reset pin ($\overline{\text{RST}}$) resets the microcontroller including the core and all the on-chip peripherals except the JTAG TAP controller (see “JTAG Interface” on page 182). The external reset sequence is as follows:

1. The external reset pin ($\overline{\text{RST}}$) is asserted for the duration specified by T_{MIN} and then de-asserted (see “Reset” on page 1239).
2. The internal reset is released and the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

To improve noise immunity and/or to delay reset at power up, the $\overline{\text{RST}}$ input may be connected to an RC network as shown in Figure 5-2 on page 197.

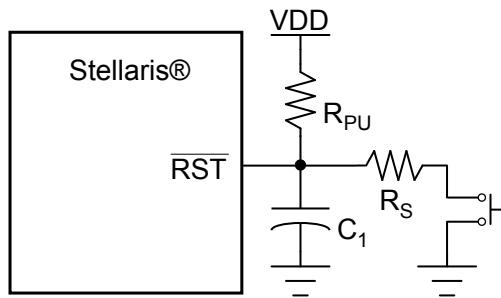
Figure 5-2. External Circuitry to Extend Power-On Reset



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

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

If the application requires the use of an external reset switch, Figure 5-3 on page 198 shows the proper circuitry to use.

Figure 5-3. Reset Circuit Controlled by Switch

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

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

$C_1 = 10\text{ nF}$

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

The external reset timing is shown in Figure 26-5 on page 1240.

5.2.2.4 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. The default condition is to generate an interrupt, so BOR must be enabled. Brown-out resets are controlled with the **Power-On and Brown-Out Reset Control (PBORCTL)** register. The **BORIOR** bit in the **PBORCTL** register must be set for a brown-out condition to trigger a reset; if **BORIOR** is clear, an interrupt is generated. 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 brown-out reset sequence is as follows:

1. When V_{DD} drops below V_{BTH} , an internal BOR condition is set.
2. If the BOR condition exists, an internal reset is asserted.
3. The internal reset is released and the microcontroller fetches and loads the initial stack pointer, the initial program counter, the first instruction designated by the program counter, and begins execution.
4. The internal BOR condition is reset after 500 μs to prevent another BOR condition from being set before software has a chance to investigate the original cause.

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 26-7 on page 1240.

5.2.2.5 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, page 294). 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 211).

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

1. A software microcontroller reset is initiated by setting the **SYSRESREQ** bit.
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 26-8 on page 1241.

5.2.2.6 Watchdog Timer Reset

The Watchdog Timer module's function is to prevent system hangs. The LM3S9L97 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 Timers” on page 525.

The watchdog reset timing is shown in Figure 26-9 on page 1241.

5.2.2.7 Special Considerations for Reset when Debugging

The Serial Wire JTAG Debug Port (SWJ-DP) is reset by any reset condition. These reset types include:

- Watchdog reset
- Brown-out reset

- Software reset
- Reset pin assertion
- Main oscillator fail

Normal operation of the device is not affected by the reset of the SWJ-DP, however, users should bear this functionality in mind during development and debugging of applications. Because some ARM debuggers expect to maintain connectivity when a system reset is requested, the SYSRESREQ should not be used when using these debuggers; instead VECTRESET, which only resets the core, should be used so that debug connectivity is uninterrupted. VECTRESET does not reset on-chip peripherals, which must be reset with specific reset operations.

If a debugger does a SYSRESREQ, or if the debugger is being used in a session and a system reset occurs due to one of the reset sources above, then the debugger loses its state, including breakpoints, watchpoints, vector catch, and trace. Most debuggers attempt a recovery, usually after reporting the error to the user. If the debugger is able to recover control, the state of the application at that time reflects that the code has been running from reset and has not stopped on any breakpoints. If the application has breakpoint instructions physically in the code, such as for system calls that run through the debugger, then the code will have entered the fault handlers.

When debugging code that requires a software reset, the SYSRESREQ software reset mechanism in the NVIC (which is used by the Stellaris Peripheral Driver Library SysCtlReset() and ROM_SysCtlReset() APIs) should not be used; instead, use the sequence of register writes with a VECTRESET in the NVIC as shown in the code below.

In addition, the ROM is mapped into address 0x0 during reset. The ROM code determines if boot loading is needed, and if not, transfers control to the normal application in Flash memory. As a result, the ROM is visible to the debugger on the reset entry. Debugging can be affected during Flash memory verification because the debugger compares the expected image with the ROM contents and not the Flash memory as intended. The disassembly shown to the user is also affected. To avoid these issues, debuggers must switch off the ROM mapping. However, if the debugger in use does not switch off the ROM, the user can either step through the first assembly instructions until the ROM gets remapped or write a 1 to the BA bit in the **ROM Control (ROMCTL)** register at location 0x400F.E0F0 using the debugger GUI, debugger command line, or debugger startup script.

Use of any reset source listed above other than software reset causes the debugger to lose connectivity.

```
//  
// Disable processor interrupts.  
//  
IntMasterDisable();  
  
//  
// Disable the PLL and the system clock divider (this is a NOP if they are  
// already disabled).  
//  
HWREG(SYSCTL_RCC) = ((HWREG(SYSCTL_RCC) & ~(SYSCTL_RCC_USESYSDIV)) |  
                     SYSCTL_RCC_BYPASS);  
HWREG(SYSCTL_RCC2) |= SYSCTL_RCC2_BYPASS2;  
  
//  
// Now, write RCC and RCC2 to their reset values.  
//
```

```

HWREG(SYSCTL_RCC) = 0x078e3ad0 | (HWREG(SYSCTL_RCC) & SYSCTL_RCC_MOSCDIS);
HWREG(SYSCTL_RCC2) = 0x07806810;
HWREG(SYSCTL_RCC) = 0x078e3ad1;

//
// Reset the deep sleep clock configuration register.
//
HWREG(SYSCTL_DSLPCLKCFG) = 0x07800000;

//
// Reset the clock gating registers.
//
HWREG(SYSCTL_RCGC0) = 0x00000040;
HWREG(SYSCTL_RCGC1) = 0;
HWREG(SYSCTL_RCGC2) = 0;
HWREG(SYSCTL_SCGC0) = 0x00000040;
HWREG(SYSCTL_SCGC1) = 0;
HWREG(SYSCTL_SCGC2) = 0;
HWREG(SYSCTL_DCGC0) = 0x00000040;
HWREG(SYSCTL_DCGC1) = 0;
HWREG(SYSCTL_DCGC2) = 0;

//
// Reset the remaining SysCtl registers.
//
HWREG(SYSCTL_PBORCTL) = 0;
HWREG(SYSCTL_IMC) = 0;
HWREG(SYSCTL_GPIOHBCTL) = 0;
HWREG(SYSCTL莫斯CCTL) = 0;
HWREG(SYSCTL_PIOSCCAL) = 0;
HWREG(SYSCTL_I2SMCLKCFG) = 0;

//
// Reset the peripherals.
//
HWREG(SYSCTL_SRCR0) = 0xffffffff;
HWREG(SYSCTL_SRCR1) = 0xffffffff;
HWREG(SYSCTL_SRCR2) = 0xffffffff;
HWREG(SYSCTL_SRCR0) = 0;
HWREG(SYSCTL_SRCR1) = 0;
HWREG(SYSCTL_SRCR2) = 0;

//
// Clear any pending SysCtl interrupts.
//
HWREG(SYSCTL_MISC) = 0xffffffff;

//
// Wait for any pending flash operations to complete.
//
while((HWREG(FLASH_FMC) & 0xffff) != 0)
{

```

```

        }
        while((HWREG(FLASH_FMC2) & 0xffff) != 0)
        {
        }

        //
        // Reset the flash controller registers.
        //
        HWREG(FLASH_FMA) = 0;
        HWREG(FLASH_FCIM) = 0;
        HWREG(FLASH_FCMISC) = 0xffffffff;
        HWREG(FLASH_FWVAL) = 0;

        //
        // Issue the core reset.
        //
        HWREG(NVIC_APINT) = NVIC_APINT_VECTKEY | NVIC_APINT_VECT_RESET;
    
```

5.2.3 Non-Maskable Interrupt

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

- The assertion of the `NMI` signal
- A main oscillator verification error
- The `NMISET` bit in the **Interrupt Control and State (INTCTRL)** register in the Cortex-M3 (see page 147).

Software must check the cause of the interrupt in order to distinguish among the sources.

5.2.3.1 NMI Pin

The `NMI` signal is the alternate function for GPIO port pin `PB7`. 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 423. 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 461. The active sense of the `NMI` signal is High; asserting the enabled `NMI` signal above V_{IH} initiates the NMI interrupt sequence.

5.2.3.2 Main Oscillator Verification Failure

The LM3S9L97 microcontroller provides a main oscillator verification circuit that generates an error condition if the oscillator is running too fast or two 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 210.

5.2.4 Power Control

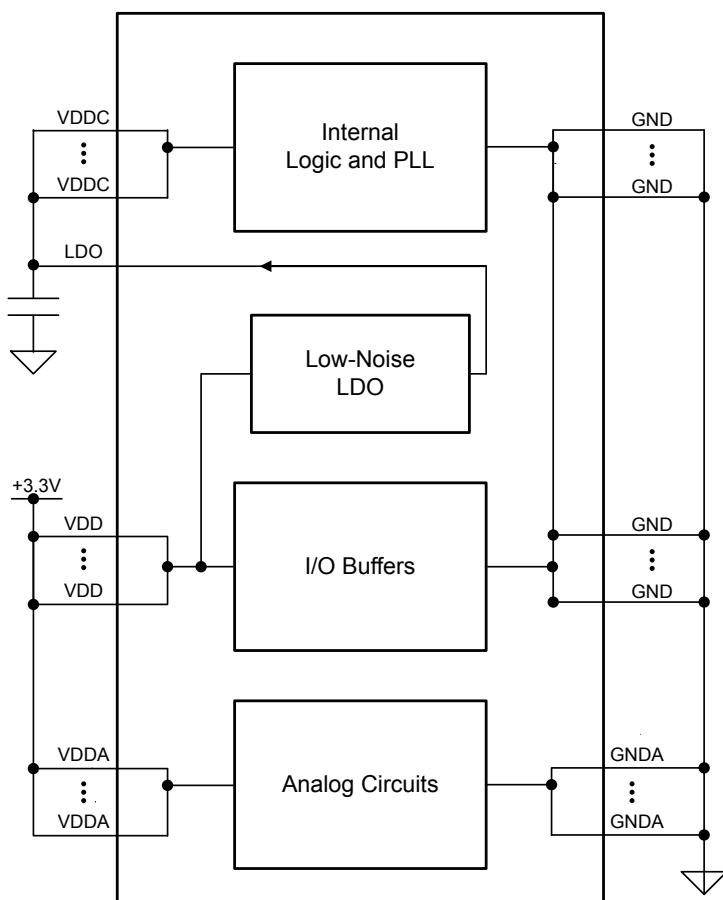
The Stellaris® microcontroller provides an integrated LDO regulator that is used to provide power to the majority of the microcontroller's internal logic. For power reduction, a non-programmable LDO may be used to scale the microcontroller's 3.3 V input voltage to 1.2V. The voltage output has a minimum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

Figure 5-4 shows the power architecture.

Note: On the printed circuit board, use the LDO output as the source of VDDC input. Do not use an external regulator to supply the voltage to VDDC. In addition, the LDO requires decoupling capacitors. See "On-Chip Low Drop-Out (LDO) Regulator Characteristics" on page 1233.

VDDA must be supplied with 3.3 V, or the microcontroller does not function properly. VDDA is the supply for all of the analog circuitry on the device, including the clock circuitry.

Figure 5-4. Power Architecture



5.2.5 Clock Control

System control determines the control of clocks in this part.

5.2.5.1 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 that is the clock source the microcontroller uses during and following POR. It does not require the use of any external components and provides a clock that is $16\text{ MHz} \pm 1\%$ at room temperature and $\pm 3\%$ across temperature. The PIOSC allows for a reduced system cost in applications that require an accurate clock source. 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. If the Hibernation Module clock source is a 32.768-kHz oscillator, the precision internal oscillator can be trimmed by software based on a reference clock for increased accuracy.
- **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 to 16.384 MHz (inclusive). If the PLL is not being used, the crystal may be any one of the supported frequencies between 1 MHz to 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 226). Note that the MOSC must have a clock source for the USB PLL.
- **Internal 30-kHz Oscillator.** The internal 30-kHz oscillator provides an operational frequency of $30\text{ kHz} \pm 50\%$. It is intended for use during Deep-Sleep power-saving modes. This power-savings mode benefits from reduced internal switching and also allows the MOSC and PIOSC to be powered down.
- **Hibernation Module Clock Source.** The Hibernation module can be clocked in one of two ways. The first way is a 4.194304-MHz crystal connected to the XOSC0 and XOSC1 pins. This clock signal is divided by 128 internally to produce the 32.768-kHz clock reference. The second way is a 32.768-kHz oscillator connected to the XOSC0 pin. The 32.768-kHz oscillator can be used for the system clock, thus eliminating the need for an additional crystal or oscillator. The Hibernation module clock source is intended to provide the system with a real-time clock source and may also provide an accurate source of Deep-Sleep or Hibernate mode power savings.

The internal system clock (SysClk), is derived from any of the above sources plus two others: the output of the main internal PLL and the precision internal oscillator divided by four ($4\text{ MHz} \pm 1\%$). The frequency of the PLL clock reference must be in the range of 3.579545 MHz to 16.384 MHz (inclusive). Table 5-4 on page 204 shows how the various clock sources can be used in a system.

Table 5-4. Clock Source Options

Clock Source	Drive PLL?		Used as SysClk?	
Precision Internal Oscillator	Yes	BYPASS = 0, OSCSRC = 0x1	Yes	BYPASS = 1, OSCSRC = 0x1
Precision Internal Oscillator divide by 4 ($4\text{ MHz} \pm 1\%$)	No	-	Yes	BYPASS = 1, OSCSRC = 0x2
Main Oscillator	Yes	BYPASS = 0, OSCSRC = 0x0	Yes	BYPASS = 1, OSCSRC = 0x0
Internal 30-kHz Oscillator	No	-	Yes	BYPASS = 1, OSCSRC = 0x3
Hibernation Module 32.768-kHz Oscillator	No	-	Yes	BYPASS = 1, OSCSRC2 = 0x7
Hibernation Module 4.194304-MHz Crystal	No	-	No	-

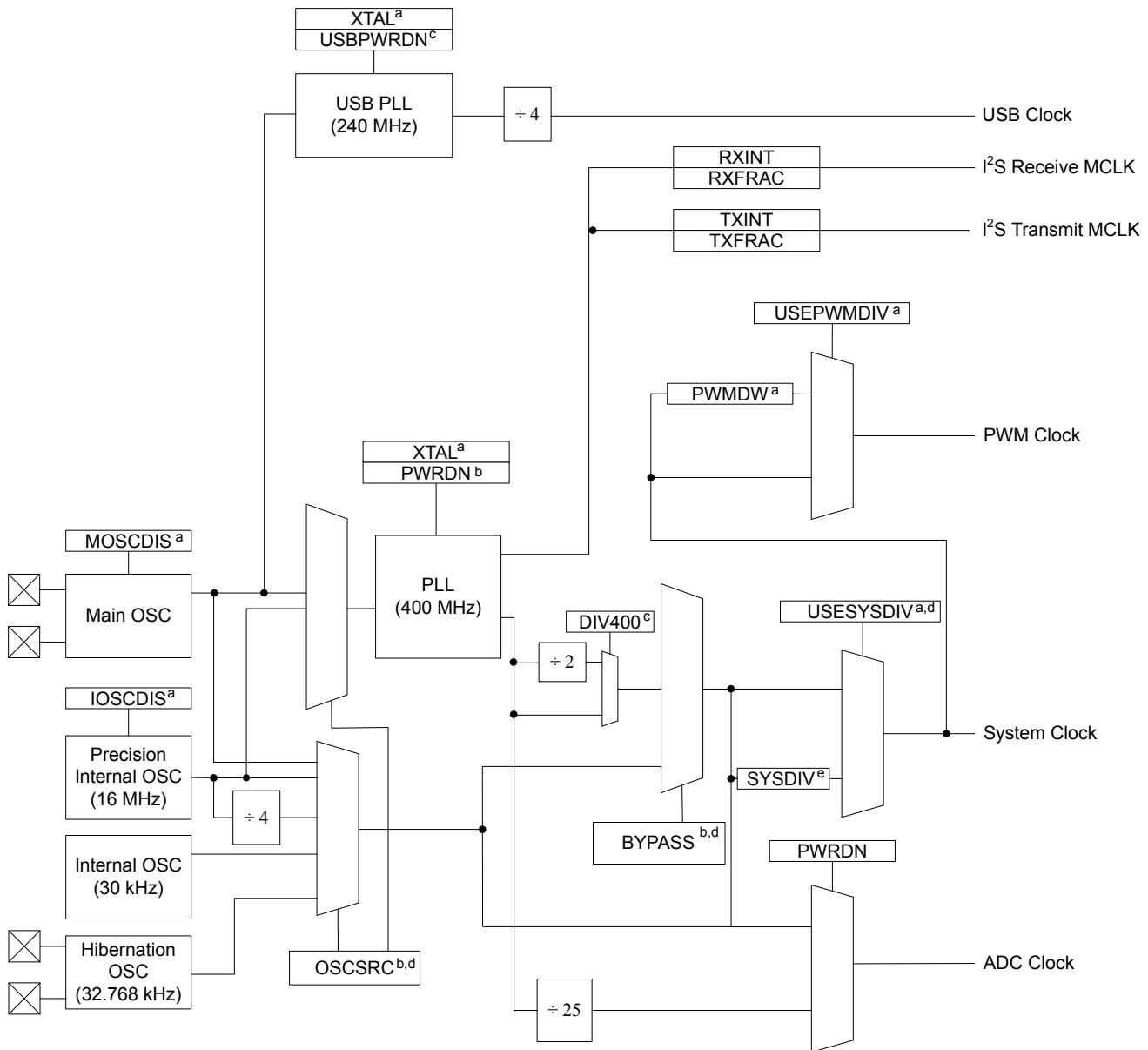
5.2.5.2 Clock Configuration

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. These registers control the following clock functionality:

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

Figure 5-5 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**).

Note: When the ADC module is in operation, the system clock must be at least 16 MHz. When the USB module is in operation, MOSC must be provided with a clock source, and the system clock must be at least 20 MHz.

Figure 5-5. 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. Control provided by **RCC** register **SYSDIV** field, **RCC2** register **SYSDIV2** field if overridden with **USERCC2** bit, or [SYSDIV2,SYSDIV2LSB] if both **USERCC2** and **DIV400** bits are set.

Note: The figure above shows all features available on all Stellaris® Tempest-class microcontrollers. Not all peripherals may be available on this device.

Using the SYSDIV and SYSDIV2 Fields

In the **RCC** register, the **SYSDIV** field specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the **BYPASS** bit in this register

is configured). When using the PLL, the VCO frequency of 400 MHz is predivided by 2 before the divisor is applied. Table 5-5 shows how the SYSDIV encoding affects the system clock frequency, depending on whether the PLL is used (BYPASS=0) or another clock source is used (BYPASS=1). The divisor is equivalent to the SYSDIV encoding plus 1. For a list of possible clock sources, see Table 5-4 on page 204.

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

SYSDIV	Divisor	Frequency (BYPASS=0)	Frequency (BYPASS=1)	StellarisWare Parameter ^a
0x0	/1	reserved	Clock source frequency/2	SYSCTL_SYSDIV_1 ^b
0x1	/2	reserved	Clock source frequency/2	SYSCTL_SYSDIV_2
0x2	/3	66.67 MHz	Clock source frequency/3	SYSCTL_SYSDIV_3
0x3	/4	50 MHz	Clock source frequency/4	SYSCTL_SYSDIV_4
0x4	/5	40 MHz	Clock source frequency/5	SYSCTL_SYSDIV_5
0x5	/6	33.33 MHz	Clock source frequency/6	SYSCTL_SYSDIV_6
0x6	/7	28.57 MHz	Clock source frequency/7	SYSCTL_SYSDIV_7
0x7	/8	25 MHz	Clock source frequency/8	SYSCTL_SYSDIV_8
0x8	/9	22.22 MHz	Clock source frequency/9	SYSCTL_SYSDIV_9
0x9	/10	20 MHz	Clock source frequency/10	SYSCTL_SYSDIV_10
0xA	/11	18.18 MHz	Clock source frequency/11	SYSCTL_SYSDIV_11
0xB	/12	16.67 MHz	Clock source frequency/12	SYSCTL_SYSDIV_12
0xC	/13	15.38 MHz	Clock source frequency/13	SYSCTL_SYSDIV_13
0xD	/14	14.29 MHz	Clock source frequency/14	SYSCTL_SYSDIV_14
0xE	/15	13.33 MHz	Clock source frequency/15	SYSCTL_SYSDIV_15
0xF	/16	12.5 MHz (default)	Clock source frequency/16	SYSCTL_SYSDIV_16

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

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

The SYSDIV2 field in the **RCC2** register is 2 bits wider than the SYSDIV field in the **RCC** register so that additional larger divisors up to /64 are possible, allowing a lower system clock frequency for improved Deep Sleep power consumption. When using the PLL, the VCO frequency of 400 MHz is predivided by 2 before the divisor is applied. The divisor is equivalent to the SYSDIV2 encoding plus 1. Table 5-6 shows how the SYSDIV2 encoding affects the system clock frequency, depending on whether the PLL is used (BYPASS2=0) or another clock source is used (BYPASS2=1). For a list of possible clock sources, see Table 5-4 on page 204.

Table 5-6. Examples of Possible System Clock Frequencies Using the SYSDIV2 Field

SYSDIV2	Divisor	Frequency (BYPASS2=0)	Frequency (BYPASS2=1)	StellarisWare Parameter ^a
0x00	/1	reserved	Clock source frequency/2	SYSCTL_SYSDIV_1 ^b
0x01	/2	reserved	Clock source frequency/2	SYSCTL_SYSDIV_2
0x02	/3	66.67 MHz	Clock source frequency/3	SYSCTL_SYSDIV_3
0x03	/4	50 MHz	Clock source frequency/4	SYSCTL_SYSDIV_4
0x09	/5	40 MHz	Clock source frequency/5	SYSCTL_SYSDIV_5
...
0x09	/10	20 MHz	Clock source frequency/10	SYSCTL_SYSDIV_10
...

**Table 5-6. Examples of Possible System Clock Frequencies Using the SYSDIV2 Field
(continued)**

SYSDIV2	Divisor	Frequency (BYPASS2=0)	Frequency (BYPASS2=1)	StellarisWare Parameter ^a
0x3F	/64	3.125 MHz	Clock source frequency/64	SYSCTL_SYSDIV_64

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

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

To allow for additional frequency choices when using the PLL, the DIV400 bit is provided along with the SYSDIV2LSB bit. When the DIV400 bit is set, bit 22 becomes the LSB for SYSDIV2. In this situation, the divisor is equivalent to the (SYSDIV2 encoding with SYSDIV2LSB appended) plus one. Table 5-7 shows the frequency choices when DIV400 is set. When the DIV400 bit is clear, SYSDIV2LSB is ignored, and the system clock frequency is determined as shown in Table 5-6 on page 207.

Table 5-7. Examples of Possible System Clock Frequencies with DIV400=1

SYSDIV2	SYSDIV2LSB	Divisor	Frequency (BYPASS2=0) ^a	StellarisWare Parameter ^b
0x00	reserved	/2	reserved	-
0x01	0	/3	reserved	-
	1	/4	reserved	-
0x02	0	/5	80 MHz	SYSCTL_SYSDIV_2_5
	1	/6	66.67 MHz	SYSCTL_SYSDIV_3
0x03	0	/7	reserved	-
	1	/8	50 MHz	SYSCTL_SYSDIV_4
0x04	0	/9	44.44 MHz	SYSCTL_SYSDIV_4_5
	1	/10	40 MHz	SYSCTL_SYSDIV_5
...
0x3F	0	/127	3.15 MHz	SYSCTL_SYSDIV_63_5
	1	/128	3.125 MHz	SYSCTL_SYSDIV_64

a. Note that DIV400 and SYSDIV2LSB are only valid when BYPASS2=0.

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

5.2.5.3 Precision Internal Oscillator Operation (PIOSC)

The microcontroller powers up with the PIOSC running. If another clock source is desired, the PIOSC should remain enabled as it is used for internal functions. The PIOSC should only be disabled during Deep-Sleep mode. It 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 three ways:

- Default calibration: clear the UTEN bit and set the UPDATE bit in the **Precision Internal Oscillator Calibration (PIOSCAL)** 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.

- Automatic calibration using the enable 32.768-kHz oscillator from the Hibernation module: set the CAL bit; the results of the calibration are shown in the RESULT field in the **Precision Internal Oscillator Statistic (PIOSCSTAT)** register. After calibration is complete, the PIOSC is trimmed using trimmed value returned in the CT field.

5.2.5.4 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 226) describes the available crystal choices and default programming values.

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

5.2.5.5 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. The PLL operates at 400 MHz, but is divided by two prior to the application of the output divisor, unless the DIV400 bit in the **RCC2** register is set.

To configure the PIOSC to be the clock source for the main PLL, program the OSCRC2 field in the **Run-Mode Clock Configuration 2 (RCC2)** register to be 0x1.

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 231). The internal translation provides a translation within $\pm 1\%$ of the targeted PLL VCO frequency. Table 26-11 on page 1236 shows the actual PLL frequency and error for a given crystal choice.

The Crystal Value field (XTAL) in the **Run-Mode Clock Configuration (RCC)** register (see page 226) describes the available crystal choices and default programming of the **PLLCFG** register. Any time the XTAL field changes, the new settings are translated and the internal PLL settings are updated.

5.2.5.6 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.

5.2.5.7 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 226 and page 234).

5.2.5.8 PLL Operation

If a PLL configuration is changed, the PLL output frequency is unstable until it reconverges (relocks) to the new setting. The time between the configuration change and relock is T_{READY} (see Table 26-10 on page 1236). 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_{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 μ 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_{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_{READY} time met), after which it changes to the PLL. Software can use many methods to ensure that the system is clocked from the main PLL, including periodically polling the `PLLRI` bit in the **Raw Interrupt Status (RIS)** register, and enabling the PLL Lock interrupt.

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

5.2.5.9 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 (PIOOSC) is disabled, it is enabled.
3. The system clock is switched from the main oscillator to the PIOOSC.
4. An internal power-on reset is initiated that lasts for 32 PIOOSC periods.
5. Reset is de-asserted and the processor is directed to the NMI handler during the reset sequence.

5.2.6 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. These registers are located in the System Control register map starting at offsets 0x600, 0x700, and 0x800, respectively.

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

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

- Run mode
- Sleep mode
- Deep-Sleep mode
- Hibernation mode

The following sections describe the different modes in detail.

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

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

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

5.2.6.1 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.

5.2.6.2 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 “Power Management” on page 111 for more details.

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

5.2.6.3 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 setting the SLEEPDEEP bit in the **System Control (SYSCTRL)** register (see page 153) and then executing a WFI instruction. Any properly configured interrupt event in the system brings the processor back into Run mode. See “Power Management” on page 111 for more details.

The Cortex-M3 processor core and the memory subsystem are not clocked in Deep-Sleep mode. Peripherals are clocked that are enabled in the **DCCGn** registers when auto-clock gating is enabled (see the **RCC** register) or the **RCGCr** registers when auto-clock gating is disabled. The system clock source is specified in the **DSLPCLKCFG** register. When the **DSLPCLKCFG** register is used, the internal oscillator source is powered up, if necessary, and other clocks are 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 **DSLPCLKCFG** 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 238.

5.2.6.4 Hibernation Mode

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

5.3 Initialization and Configuration

The PLL is configured using direct register writes to the **RCC/RCC2** register. If the **RCC2** register is being used, the **USERRCC2** 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 **PLLLOCKED** bit in the **Raw Interrupt Status (RIS)** register.
5. Enable use of the PLL by clearing the **BYPASS** bit in **RCC/RCC2**.

5.4 Register Map

Table 5-8 on page 213 lists the System Control registers, grouped by function. The offset listed is a hexadecimal increment to the register's address, relative to the System Control base address of 0x400F.E000.

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

Additional Flash and ROM registers defined in the System Control register space are described in the “Internal Memory” on page 328.

Table 5-8. System Control Register Map

Offset	Name	Type	Reset	Description	See page
0x000	DID0	RO	-	Device Identification 0	215
0x004	DID1	RO	-	Device Identification 1	245
0x008	DC0	RO	0x00BF.003F	Device Capabilities 0	247
0x010	DC1	RO	-	Device Capabilities 1	248
0x014	DC2	RO	0x130F.5337	Device Capabilities 2	250
0x018	DC3	RO	0xBFFF.8FFF	Device Capabilities 3	252
0x01C	DC4	RO	0x5104.F1FF	Device Capabilities 4	254
0x020	DC5	RO	0x0F30.003F	Device Capabilities 5	256
0x024	DC6	RO	0x0000.0013	Device Capabilities 6	258
0x028	DC7	RO	0xFFFF.FFFF	Device Capabilities 7	259
0x02C	DC8	RO	0xFFFF.FFFF	Device Capabilities 8 ADC Channels	263
0x030	PBORCTL	R/W	0x0000.7FFD	Brown-Out Reset Control	217
0x040	SRCR0	R/W	0x00000000	Software Reset Control 0	294
0x044	SRCR1	R/W	0x00000000	Software Reset Control 1	296
0x048	SRCR2	R/W	0x00000000	Software Reset Control 2	299
0x050	RIS	RO	0x0000.0000	Raw Interrupt Status	218
0x054	IMC	R/W	0x0000.0000	Interrupt Mask Control	220
0x058	MISC	R/W1C	0x0000.0000	Masked Interrupt Status and Clear	222
0x05C	RESC	R/W	-	Reset Cause	224
0x060	RCC	R/W	0x078E.3AD1	Run-Mode Clock Configuration	226
0x064	PLLCFG	RO	-	XTAL to PLL Translation	231
0x06C	GPIOHBCTL	R/W	0x0000.0000	GPIO High-Performance Bus Control	232
0x070	RCC2	R/W	0x07C0.6810	Run-Mode Clock Configuration 2	234
0x07C	MOSCCTL	R/W	0x0000.0000	Main Oscillator Control	237
0x100	RCGC0	R/W	0x00000040	Run Mode Clock Gating Control Register 0	268
0x104	RCGC1	R/W	0x00000000	Run Mode Clock Gating Control Register 1	276
0x108	RCGC2	R/W	0x00000000	Run Mode Clock Gating Control Register 2	285
0x110	SCGC0	R/W	0x00000040	Sleep Mode Clock Gating Control Register 0	271

Table 5-8. System Control Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x114	SCGC1	R/W	0x00000000	Sleep Mode Clock Gating Control Register 1	279
0x118	SCGC2	R/W	0x00000000	Sleep Mode Clock Gating Control Register 2	288
0x120	DCGC0	R/W	0x00000040	Deep Sleep Mode Clock Gating Control Register 0	274
0x124	DCGC1	R/W	0x00000000	Deep-Sleep Mode Clock Gating Control Register 1	282
0x128	DCGC2	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 2	291
0x144	DSPCLKCFG	R/W	0x0780.0000	Deep Sleep Clock Configuration	238
0x150	PIOSCCAL	R/W	0x0000.0000	Precision Internal Oscillator Calibration	240
0x154	PIOSCSTAT	RO	0x0000.0040	Precision Internal Oscillator Statistics	242
0x170	I2SMCLKCFG	R/W	0x0000.0000	I2S MCLK Configuration	243
0x190	DC9	RO	0x00FF.00FF	Device Capabilities 9 ADC Digital Comparators	265
0x1A0	NVMSTAT	RO	0x0000.0001	Non-Volatile Memory Information	267

5.5 Register Descriptions

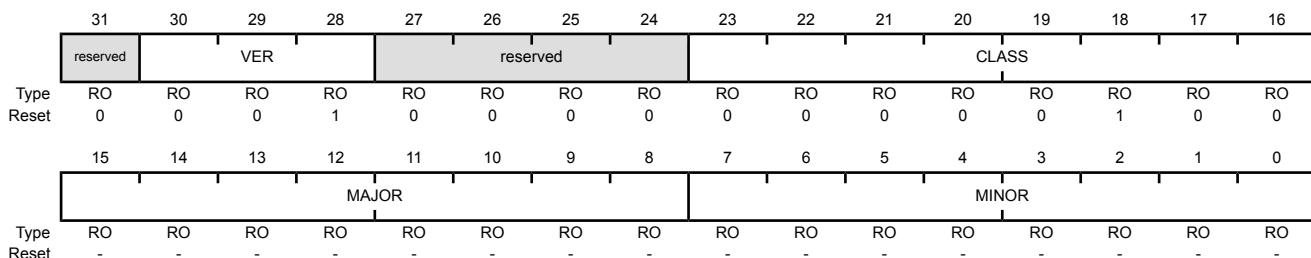
All addresses given are relative to the System Control base address of 0x400F.E000.

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

This register identifies the version of the microcontroller.

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 DID0 register format version. The version number is numeric. The value of the VER field is encoded as follows (all other encodings are reserved):</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x1</td> <td>Second version of the DID0 register format.</td> </tr> </tbody> </table>	Value	Description	0x1	Second version of the DID0 register format.
Value	Description							
0x1	Second version of the DID0 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 CLASS field value identifies the internal design from which all mask sets are generated for all microcontrollers in a particular product line. The CLASS field value is changed for new product lines, for changes in fab process (for example, a remap or shrink), or any case where the MAJOR or MINOR fields require differentiation from prior microcontrollers. The value of the CLASS field is encoded as follows (all other encodings are reserved):</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>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							

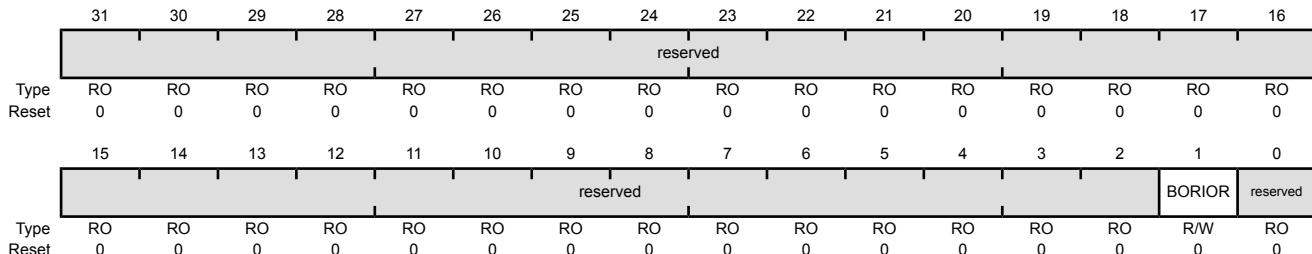
Bit/Field	Name	Type	Reset	Description								
15:8	MAJOR	RO	-	<p>Major Revision</p> <p>This field specifies the major revision number of the microcontroller. The major revision reflects changes to base layers of the design. The major revision number is indicated in the part number as a letter (A for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

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

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

Brown-Out Reset Control (PBORCTL)

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



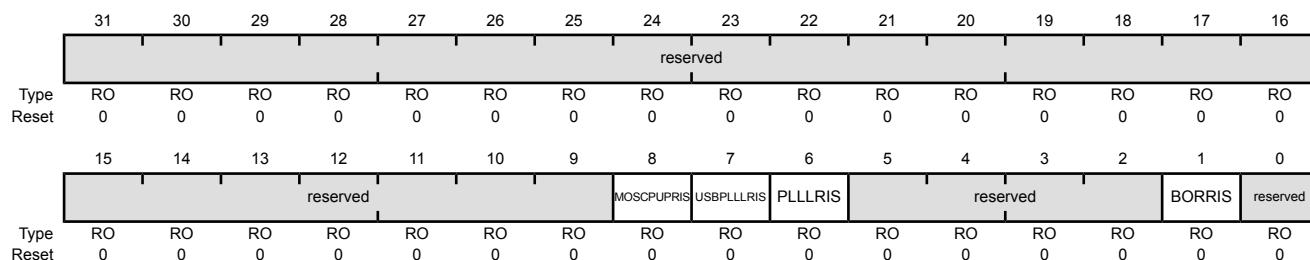
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORIOR	R/W	0	BOR Interrupt or Reset
		Value	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



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	MOSC Power Up Raw Interrupt Status
	Value	Description		
	1	Sufficient time has passed for the MOSC to reach the expected frequency. The value for this power-up time is indicated by T_{MOSC_SETTLE} .		
	0	Sufficient time has not passed for the MOSC to reach the expected frequency.		
	This bit is cleared by writing a 1 to the MOSCPUPMIS bit in the MISC register.			
7	USBPLLRISS	RO	0	USB PLL Lock Raw Interrupt Status
	Value	Description		
	1	The USB PLL timer has reached T_{READY} indicating that sufficient time has passed for the USB PLL to lock.		
	0	The USB PLL timer has not reached T_{READY} .		
	This bit is cleared by writing a 1 to the USBPLLLMIS bit in the MISC register.			
6	PLLLRIS	RO	0	PLL Lock Raw Interrupt Status
	Value	Description		
	1	The PLL timer has reached T_{READY} indicating that sufficient time has passed for the PLL to lock.		
	0	The PLL timer has not reached T_{READY} .		
	This bit is cleared by writing a 1 to the PLLLMIS bit in the MISC register.			

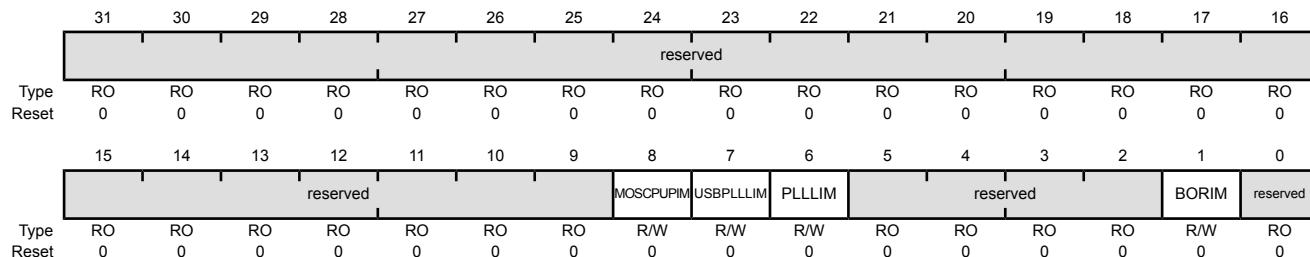
Bit/Field	Name	Type	Reset	Description						
5:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
1	BORRIS	RO	0	<p>Brown-Out Reset Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>A brown-out condition is currently active.</td></tr> <tr> <td>0</td><td>A brown-out condition is not currently active.</td></tr> </tbody> </table> <p>Note the BORIOR bit in the PBORCTL register must be cleared to cause an interrupt due to a Brown Out Event. This bit is cleared by writing a 1 to the BORMIS bit in the MISC register.</p>	Value	Description	1	A brown-out condition is currently active.	0	A brown-out condition is not currently active.
Value	Description									
1	A brown-out condition is currently active.									
0	A brown-out condition is not currently active.									
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



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	MOSC Power Up Interrupt Mask
		Value	Description	
		1	An interrupt is sent to the interrupt controller when the MOSCPUPRIS bit in the RIS register is set.	
		0	The MOSCPUPRIS interrupt is suppressed and not sent to the interrupt controller.	
7	USBPLLIM	R/W	0	USB PLL Lock Interrupt Mask
		Value	Description	
		1	An interrupt is sent to the interrupt controller when the USBPLLRLIS bit in the RIS register is set.	
		0	The USBPLLRLIS interrupt is suppressed and not sent to the interrupt controller.	
6	PLLIM	R/W	0	PLL Lock Interrupt Mask
		Value	Description	
		1	An interrupt is sent to the interrupt controller when the PLLLRIS bit in the RIS register is set.	
		0	The PLLLRIS 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.

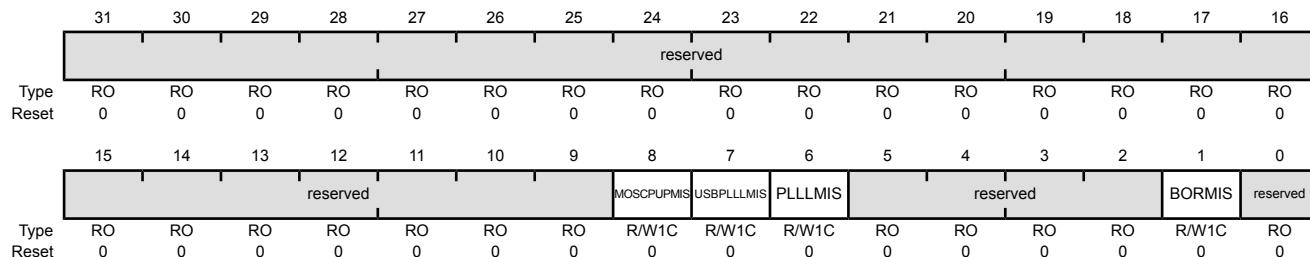
Bit/Field	Name	Type	Reset	Description
1	BORIM	R/W	0	Brown-Out Reset Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the BORRIS bit in the RIS register is set. 0 The BORRIS interrupt is suppressed and not sent to the interrupt controller.
0	reserved	RO	0	Software 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 218).

Masked Interrupt Status and Clear (MISC)

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



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
	Value	Description		
	1	When read, a 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the MOSC PLL to lock. Writing a 1 to this bit clears it and also the MOSCPUPRIS bit in the RIS register.		
	0	When read, a 0 indicates that sufficient time has not passed for the MOSC PLL to lock. A write of 0 has no effect on the state of this bit.		
7	USBPLLIMIS	R/W1C	0	USB 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 USB PLL to lock. Writing a 1 to this bit clears it and also the USBPLLISRIS bit in the RIS register.		
	0	When read, a 0 indicates that sufficient time has not passed for the USB PLL to lock. A write of 0 has no effect on the state of this bit.		

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 RIS 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 RIS 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 a 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	R/W				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-				
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
	reserved														WDT1	SW	WDTO	BOR	POR	EXT
Type	RO	RO	RO	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	-	-	-	-	-	-	-	-	-	

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.	
15:6	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	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> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>When read, this bit indicates that a software reset has caused a reset event.</td></tr> <tr> <td>0</td><td>When read, this bit indicates that a software reset has not generated a reset since the previous power-on reset. Writing a 0 to this bit clears it.</td></tr> </tbody> </table>	Value	Description	1	When read, this bit indicates that a software reset has caused a reset event.	0	When read, this bit indicates that a software reset has not generated a reset since the previous power-on reset. Writing a 0 to this bit clears it.
Value	Description									
1	When read, this bit indicates that a software reset has caused a reset event.									
0	When read, this bit indicates that a software reset has not generated a reset since the previous power-on reset. Writing a 0 to this bit clears it.									
3	WDT0	R/W	-	<p>Watchdog Timer 0 Reset</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>When read, this bit indicates that Watchdog Timer 0 timed out and generated a reset.</td></tr> <tr> <td>0</td><td>When read, this bit indicates that Watchdog Timer 0 has not generated a reset since the previous power-on reset. Writing a 0 to this bit clears it.</td></tr> </tbody> </table>	Value	Description	1	When read, this bit indicates that Watchdog Timer 0 timed out and generated a reset.	0	When read, this bit indicates that Watchdog Timer 0 has not generated a reset since the previous power-on reset. Writing a 0 to this bit clears it.
Value	Description									
1	When read, this bit indicates that Watchdog Timer 0 timed out and generated a reset.									
0	When read, this bit indicates that Watchdog Timer 0 has not generated a reset since the previous power-on reset. Writing a 0 to this bit clears it.									
2	BOR	R/W	-	<p>Brown-Out Reset</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>When read, this bit indicates that a brown-out reset has caused a reset event.</td></tr> <tr> <td>0</td><td>When read, this bit indicates that a brown-out reset has not generated a reset since the previous power-on reset. Writing a 0 to this bit clears it.</td></tr> </tbody> </table>	Value	Description	1	When read, this bit indicates that a brown-out reset has caused a reset event.	0	When read, this bit indicates that a brown-out reset has not generated a reset since the previous power-on reset. Writing a 0 to this bit clears it.
Value	Description									
1	When read, this bit indicates that a brown-out reset has caused a reset event.									
0	When read, this bit indicates that a brown-out reset has not generated a reset since the previous power-on reset. Writing a 0 to this bit clears it.									
1	POR	R/W	-	<p>Power-On Reset</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>When read, this bit indicates that a power-on reset has caused a reset event.</td></tr> <tr> <td>0</td><td>When read, this bit indicates that a power-on reset has not generated a reset. Writing a 0 to this bit clears it.</td></tr> </tbody> </table>	Value	Description	1	When read, this bit indicates that a power-on reset has caused a reset event.	0	When read, this bit indicates that a power-on reset has not generated a reset. Writing a 0 to this bit clears it.
Value	Description									
1	When read, this bit indicates that a power-on reset has caused a reset event.									
0	When read, this bit indicates that a power-on reset has not generated a reset. Writing a 0 to this bit clears it.									
0	EXT	R/W	-	<p>External Reset</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>When read, this bit indicates that an external reset (\overline{RST} assertion) has caused a reset event.</td></tr> <tr> <td>0</td><td>When read, this bit indicates that an external reset (\overline{RST} assertion) has not caused a reset event since the previous power-on reset. Writing a 0 to this bit clears it.</td></tr> </tbody> </table>	Value	Description	1	When read, this bit indicates that an external reset (\overline{RST} assertion) has caused a reset event.	0	When read, this bit indicates that an external reset (\overline{RST} assertion) has not caused a reset event since the previous power-on reset. Writing a 0 to this bit clears it.
Value	Description									
1	When read, this bit indicates that an external reset (\overline{RST} assertion) has caused a reset event.									
0	When read, this bit indicates that an external reset (\overline{RST} assertion) has not caused a reset event since the previous power-on reset. Writing a 0 to this bit clears it.									

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																
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
reserved																
Type	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	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 Sleep-Mode Clock Gating Control (SCGCn) registers and Deep-Sleep-Mode Clock Gating Control (DCGCn) registers if the microcontroller enters a Sleep or Deep-Sleep mode (respectively).</p>

Value	Description
1	The SCGCn or DCGCn registers are used to control the clocks distributed to the peripherals when the microcontroller is in a sleep mode. The SCGCn and DCGCn registers allow unused peripherals to consume less power when the microcontroller is in a sleep mode.
0	The Run-Mode Clock Gating Control (RCGCr) registers are used when the microcontroller enters a sleep mode.

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

26:23	SYSDIV	R/W	0xF	<p>System Clock Divisor</p> <p>Specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the BYPASS bit in this register is configured). See Table 5-5 on page 207 for bit encodings.</p> <p>If the SYSDIV value is less than MINSYSDIV (see page 248), 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>
-------	--------	-----	-----	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Bit/Field	Name	Type	Reset	Description																		
22	USESYS DIV	R/W	0	Enable System Clock Divider																		
				<table> <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. If the USERCC2 bit in the RCC2 register is set, then the SYS DIV2 field in the RCC2 register is used as the system clock divider rather than the SYS DIV field in this register.</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. If the USERCC2 bit in the RCC2 register is set, then the SYS DIV2 field in the RCC2 register is used as the system clock divider rather than the SYS DIV field in this register.	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. If the USERCC2 bit in the RCC2 register is set, then the SYS DIV2 field in the RCC2 register is used as the system clock divider rather than the SYS DIV field in this register.																					
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	Enable PWM Clock Divisor																		
				<table> <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.																					
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> <table> <thead> <tr> <th>Value</th><th>Divisor</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>/2</td></tr> <tr> <td>0x1</td><td>/4</td></tr> <tr> <td>0x2</td><td>/8</td></tr> <tr> <td>0x3</td><td>/16</td></tr> <tr> <td>0x4</td><td>/32</td></tr> <tr> <td>0x5</td><td>/64</td></tr> <tr> <td>0x6</td><td>/64</td></tr> <tr> <td>0x7</td><td>/64 (default)</td></tr> </tbody> </table>	Value	Divisor	0x0	/2	0x1	/4	0x2	/8	0x3	/16	0x4	/32	0x5	/64	0x6	/64	0x7	/64 (default)
Value	Divisor																					
0x0	/2																					
0x1	/4																					
0x2	/8																					
0x3	/16																					
0x4	/32																					
0x5	/64																					
0x6	/64																					
0x7	/64 (default)																					
16:14	reserved	RO	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	PLL Power Down																		
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The PLL is powered down. Care must be taken to ensure that another clock source is functioning and that the BYPASS bit is set before setting this bit.</td></tr> <tr> <td>0</td><td>The PLL is operating normally.</td></tr> </tbody> </table>	Value	Description	1	The PLL is powered down. Care must be taken to ensure that another clock source is functioning and that the BYPASS bit is set before setting this bit.	0	The PLL is operating normally.												
Value	Description																					
1	The PLL is powered down. Care must be taken to ensure that another clock source is functioning and that the BYPASS bit is set before setting this bit.																					
0	The PLL is operating normally.																					

Bit/Field	Name	Type	Reset	Description
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	PLL Bypass Value Description 1 The system clock is derived from the OSC source and divided by the divisor specified by SYSDIV. 0 The system clock is the PLL output clock divided by the divisor specified by SYSDIV.

See Table 5-5 on page 207 for programming guidelines.

Note: The ADC must be clocked from the PLL or directly from a 16-MHz clock source to operate properly.

Bit/Field	Name	Type	Reset	Description																																																																								
10:6	XTAL	R/W	0x0B	<p>Crystal Value</p> <p>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 26-11 on page 1236 for more information.</p> <p>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.</p> <table> <thead> <tr> <th>Value</th> <th>Crystal Frequency (MHz) Not Using the PLL</th> <th>Crystal Frequency (MHz) Using the PLL</th> </tr> </thead> <tbody> <tr> <td>0x00</td> <td>1.000</td> <td>reserved</td> </tr> <tr> <td>0x01</td> <td>1.8432</td> <td>reserved</td> </tr> <tr> <td>0x02</td> <td>2.000</td> <td>reserved</td> </tr> <tr> <td>0x03</td> <td>2.4576</td> <td>reserved</td> </tr> <tr> <td>0x04</td> <td></td> <td>3.579545 MHz</td> </tr> <tr> <td>0x05</td> <td></td> <td>3.6864 MHz</td> </tr> <tr> <td>0x06</td> <td></td> <td>4 MHz (USB)</td> </tr> <tr> <td>0x07</td> <td></td> <td>4.096 MHz</td> </tr> <tr> <td>0x08</td> <td></td> <td>4.9152 MHz</td> </tr> <tr> <td>0x09</td> <td></td> <td>5 MHz (USB)</td> </tr> <tr> <td>0x0A</td> <td></td> <td>5.12 MHz</td> </tr> <tr> <td>0x0B</td> <td></td> <td>6 MHz (reset value)(USB)</td> </tr> <tr> <td>0x0C</td> <td></td> <td>6.144 MHz</td> </tr> <tr> <td>0x0D</td> <td></td> <td>7.3728 MHz</td> </tr> <tr> <td>0x0E</td> <td></td> <td>8 MHz (USB)</td> </tr> <tr> <td>0x0F</td> <td></td> <td>8.192 MHz</td> </tr> <tr> <td>0x10</td> <td></td> <td>10.0 MHz (USB)</td> </tr> <tr> <td>0x11</td> <td></td> <td>12.0 MHz (USB)</td> </tr> <tr> <td>0x12</td> <td></td> <td>12.288 MHz</td> </tr> <tr> <td>0x13</td> <td></td> <td>13.56 MHz</td> </tr> <tr> <td>0x14</td> <td></td> <td>14.31818 MHz</td> </tr> <tr> <td>0x15</td> <td></td> <td>16.0 MHz (USB)</td> </tr> <tr> <td>0x16</td> <td></td> <td>16.384 MHz</td> </tr> </tbody> </table>	Value	Crystal Frequency (MHz) Not Using the PLL	Crystal Frequency (MHz) Using the PLL	0x00	1.000	reserved	0x01	1.8432	reserved	0x02	2.000	reserved	0x03	2.4576	reserved	0x04		3.579545 MHz	0x05		3.6864 MHz	0x06		4 MHz (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
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 Selects the input source for the OSC. The values are:</p> <table> <thead> <tr> <th>Value</th><th>Input Source</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>MOSC Main oscillator</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 RCC2 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> <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> <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

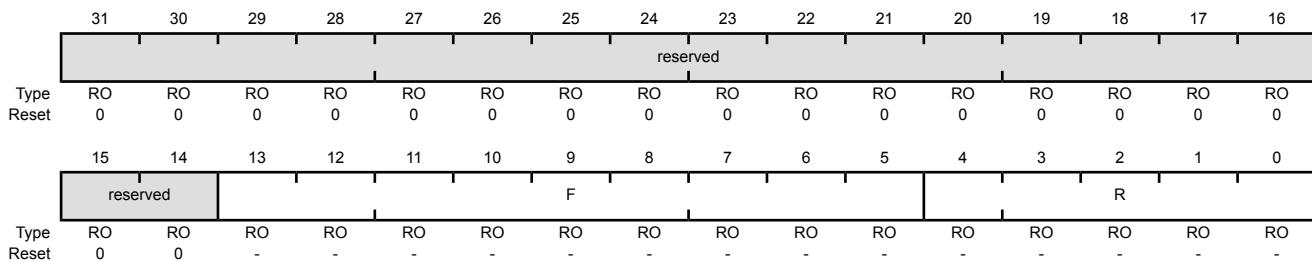
This register provides a means of translating external crystal frequencies into the appropriate PLL settings. This register is initialized during the reset sequence and updated anytime that the **XTAL** field changes in the **Run-Mode Clock Configuration (RCC)** register (see page 226).

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

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

XTAL to PLL Translation (PLLCFG)

Base 0x400F.E000
Offset 0x064
Type RO, reset -



Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	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 High-Performance 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 High-Performance 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 9-7 on page 434).

GPIO High-Performance 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 High-Performance Bus This bit defines the memory aperture for Port J. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
7	PORTH	R/W	0	Port H Advanced High-Performance Bus This bit defines the memory aperture for Port H. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
6	PORTG	R/W	0	Port G Advanced High-Performance Bus This bit defines the memory aperture for Port G. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.

Bit/Field	Name	Type	Reset	Description						
5	PORTF	R/W	0	<p>Port F Advanced High-Performance Bus This bit defines the memory aperture for Port F.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Advanced High-Performance Bus (AHB)</td></tr> <tr> <td>0</td><td>Advanced Peripheral Bus (APB). This bus is the legacy bus.</td></tr> </tbody> </table>	Value	Description	1	Advanced High-Performance Bus (AHB)	0	Advanced Peripheral Bus (APB). This bus is the legacy bus.
Value	Description									
1	Advanced High-Performance Bus (AHB)									
0	Advanced Peripheral Bus (APB). This bus is the legacy bus.									
4	PORTE	R/W	0	<p>Port E Advanced High-Performance Bus This bit defines the memory aperture for Port E.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Advanced High-Performance Bus (AHB)</td></tr> <tr> <td>0</td><td>Advanced Peripheral Bus (APB). This bus is the legacy bus.</td></tr> </tbody> </table>	Value	Description	1	Advanced High-Performance Bus (AHB)	0	Advanced Peripheral Bus (APB). This bus is the legacy bus.
Value	Description									
1	Advanced High-Performance Bus (AHB)									
0	Advanced Peripheral Bus (APB). This bus is the legacy bus.									
3	PORTD	R/W	0	<p>Port D Advanced High-Performance Bus This bit defines the memory aperture for Port D.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Advanced High-Performance Bus (AHB)</td></tr> <tr> <td>0</td><td>Advanced Peripheral Bus (APB). This bus is the legacy bus.</td></tr> </tbody> </table>	Value	Description	1	Advanced High-Performance Bus (AHB)	0	Advanced Peripheral Bus (APB). This bus is the legacy bus.
Value	Description									
1	Advanced High-Performance Bus (AHB)									
0	Advanced Peripheral Bus (APB). This bus is the legacy bus.									
2	PORTC	R/W	0	<p>Port C Advanced High-Performance Bus This bit defines the memory aperture for Port C.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Advanced High-Performance Bus (AHB)</td></tr> <tr> <td>0</td><td>Advanced Peripheral Bus (APB). This bus is the legacy bus.</td></tr> </tbody> </table>	Value	Description	1	Advanced High-Performance Bus (AHB)	0	Advanced Peripheral Bus (APB). This bus is the legacy bus.
Value	Description									
1	Advanced High-Performance Bus (AHB)									
0	Advanced Peripheral Bus (APB). This bus is the legacy bus.									
1	PORTB	R/W	0	<p>Port B Advanced High-Performance Bus This bit defines the memory aperture for Port B.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Advanced High-Performance Bus (AHB)</td></tr> <tr> <td>0</td><td>Advanced Peripheral Bus (APB). This bus is the legacy bus.</td></tr> </tbody> </table>	Value	Description	1	Advanced High-Performance Bus (AHB)	0	Advanced Peripheral Bus (APB). This bus is the legacy bus.
Value	Description									
1	Advanced High-Performance Bus (AHB)									
0	Advanced Peripheral Bus (APB). This bus is the legacy bus.									
0	PORTA	R/W	0	<p>Port A Advanced High-Performance Bus This bit defines the memory aperture for Port A.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Advanced High-Performance Bus (AHB)</td></tr> <tr> <td>0</td><td>Advanced Peripheral Bus (APB). This bus is the legacy bus.</td></tr> </tbody> </table>	Value	Description	1	Advanced High-Performance Bus (AHB)	0	Advanced Peripheral Bus (APB). This bus is the legacy bus.
Value	Description									
1	Advanced High-Performance Bus (AHB)									
0	Advanced Peripheral Bus (APB). This bus is the legacy bus.									

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

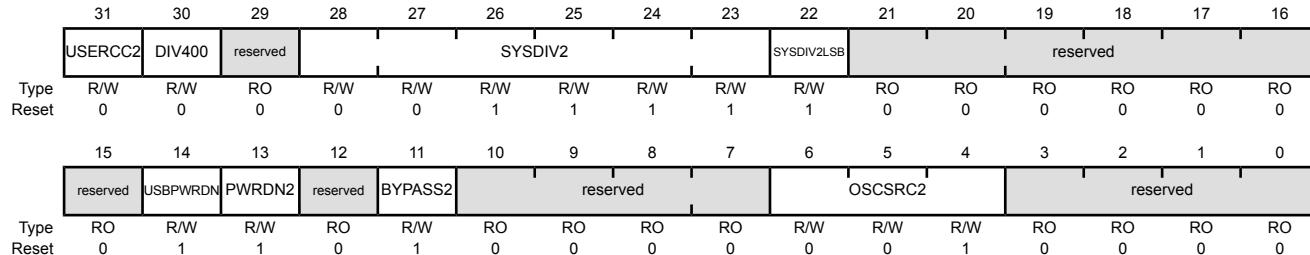
This register overrides the **RCC** equivalent register fields, as shown in Table 5-9, when the **USERCC2** bit is set, allowing the extended capabilities of the **RCC2** register to be used while also providing a means to be backward-compatible to previous parts. Each **RCC2** field that supersedes an **RCC** field is located at the same LSB bit position; however, some **RCC2** fields are larger than the corresponding **RCC** field.

Table 5-9. RCC2 Fields that Override RCC Fields

RCC2 Field...	Overrides RCC Field
SYSDIV2, bits[28:23]	SYSDIV, bits[26:23]
PWRDN2, bit[13]	PWRDN, bit[13]
BYPASS2, bit[11]	BYPASS, bit[11]
OSCSRC2, bits[6:4]	OSCSRC, bits[5:4]

Run-Mode Clock Configuration 2 (RCC2)

Base 0x400F.E000
Offset 0x070
Type R/W, reset 0x07C0.6810



Bit/Field	Name	Type	Reset	Description
31	USERCC2	R/W	0	Use RCC2
				Value Description
			1	The RCC2 register fields override the RCC register fields.
			0	The RCC register fields are used, and the fields in RCC2 are ignored.
30	DIV400	R/W	0	Divide PLL as 400 MHz vs. 200 MHz This bit, along with the SYSDIV2LSB bit, allows additional frequency choices.
				Value Description
			1	Append the SYSDIV2LSB bit to the SYSDIV2 field to create a 7 bit divisor using the 400 MHz PLL output, see Table 5-7 on page 208.
			0	Use SYSDIV2 as is and apply to 200 MHz predivided PLL output. See Table 5-6 on page 207 for programming guidelines.
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.

Bit/Field	Name	Type	Reset	Description						
28:23	SYSDIV2	R/W	0x0F	<p>System Clock Divisor 2</p> <p>Specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the BYPASS2 bit is configured). SYSDIV2 is used for the divisor when both the USESYSDIV bit in the RCC register and the USERCC2 bit in this register are set. See Table 5-6 on page 207 for programming guidelines.</p>						
22	SYSDIV2LSB	R/W	1	<p>Additional LSB for SYSDIV2</p> <p>When DIV400 is set, this bit becomes the LSB of SYSDIV2. If DIV400 is clear, this bit is not used. See Table 5-6 on page 207 for programming guidelines.</p> <p>This bit can only be set or cleared when DIV400 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> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The USB PLL is powered down.</td></tr> <tr> <td>0</td><td>The USB PLL operates normally.</td></tr> </tbody> </table>	Value	Description	1	The USB PLL is powered down.	0	The USB PLL operates normally.
Value	Description									
1	The USB PLL is powered down.									
0	The USB PLL operates normally.									
13	PWRDN2	R/W	1	<p>Power-Down PLL 2</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The PLL is powered down.</td></tr> <tr> <td>0</td><td>The PLL operates normally.</td></tr> </tbody> </table>	Value	Description	1	The PLL is powered down.	0	The PLL operates normally.
Value	Description									
1	The PLL is powered down.									
0	The PLL operates normally.									
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
11	BYPASS2	R/W	1	<p>PLL Bypass 2</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The system clock is derived from the OSC source and divided by the divisor specified by SYSDIV2.</td></tr> <tr> <td>0</td><td>The system clock is the PLL output clock divided by the divisor specified by SYSDIV2.</td></tr> </tbody> </table> <p>See Table 5-6 on page 207 for programming guidelines.</p> <p>Note: The ADC must be clocked from the PLL or directly from a 16-MHz clock source to operate properly.</p>	Value	Description	1	The system clock is derived from the OSC source and divided by the divisor specified by SYSDIV2.	0	The system clock is the PLL output clock divided by the divisor specified by SYSDIV2.
Value	Description									
1	The system clock is derived from the OSC source and divided by the divisor specified by SYSDIV2.									
0	The system clock is the PLL output clock divided by the divisor specified by SYSDIV2.									
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.						

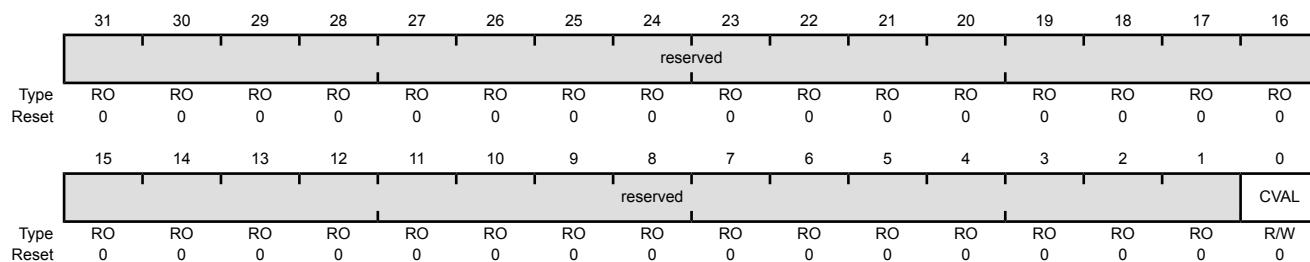
Bit/Field	Name	Type	Reset	Description														
6:4	OSCSRC2	R/W	0x1	Oscillator Source 2 Selects the input source for the OSC. The values are:														
				<table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0x0</td><td>MOSC Main oscillator</td></tr><tr><td>0x1</td><td>PIOSC Precision internal oscillator</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><tr><td>0x4-0x6</td><td>Reserved</td></tr><tr><td>0x7</td><td>32.768 kHz 32.768-kHz external oscillator</td></tr></tbody></table>	Value	Description	0x0	MOSC Main oscillator	0x1	PIOSC Precision internal oscillator	0x2	PIOSC/4 Precision internal oscillator / 4	0x3	30 kHz 30-kHz internal oscillator	0x4-0x6	Reserved	0x7	32.768 kHz 32.768-kHz external oscillator
Value	Description																	
0x0	MOSC Main oscillator																	
0x1	PIOSC Precision internal oscillator																	
0x2	PIOSC/4 Precision internal oscillator / 4																	
0x3	30 kHz 30-kHz internal oscillator																	
0x4-0x6	Reserved																	
0x7	32.768 kHz 32.768-kHz external oscillator																	
3:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.														

Register 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



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 (DSLPCLKCFG), offset 0x144

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

Deep Sleep Clock Configuration (DSLPCLKCFG)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved																
DSDIVORIDE																
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																
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	Divide Field Override 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 SYSDIV field in the RCC register or the SYSDIV2 field in the RCC2 register during Deep Sleep. This divider is applied to the source selected by the DSOSCSRC field.
	Value Description			
	0x0	/1		
	0x1	/2		
	0x2	/3		
	0x3	/4		
		
	0x3F	/64		
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 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 Use the main oscillator as the source.</td></tr> <tr> <td>0x1</td><td>PIOSC 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 Use the 30-kHz internal oscillator as the source.</td></tr> <tr> <td>0x4-0x6</td><td>Reserved</td></tr> <tr> <td>0x7</td><td>32.768 kHz Use the Hibernation module 32.768-kHz external oscillator as the source.</td></tr> </tbody> </table> <p>Note: 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.</p>	Value	Description	0x0	MOSC Use the main oscillator as the source.	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.	0x4-0x6	Reserved	0x7	32.768 kHz Use the Hibernation module 32.768-kHz external oscillator as the source.
Value	Description																	
0x0	MOSC Use the main oscillator as the source.																	
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.																	
0x4-0x6	Reserved																	
0x7	32.768 kHz Use the Hibernation module 32.768-kHz external oscillator as the source.																	
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: Precision Internal Oscillator Calibration (PIOSCCAL), offset 0x150

This register provides the ability to update or recalibrate the precision internal oscillator. Note that a 32.768-kHz oscillator must be used as the Hibernation module clock source for the user to be able to calibrate the PIOSC.

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
Type	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
Type	reserved							CAL	UPDATE	reserved	UT					
Reset	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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.
30:10	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.
9	CAL	R/W	0	Start Calibration
				Value Description
			1	Starts a new calibration of the PIOSC. Results are in the PIOSCSTAT register. The resulting trim value from the operation is active in the PIOSC after the calibration completes. The result overrides any previous update trim operation whether the calibration passes or fails.
			0	No action.
				This bit is auto-cleared when the calibration finishes.
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 PIOSCSTAT 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.

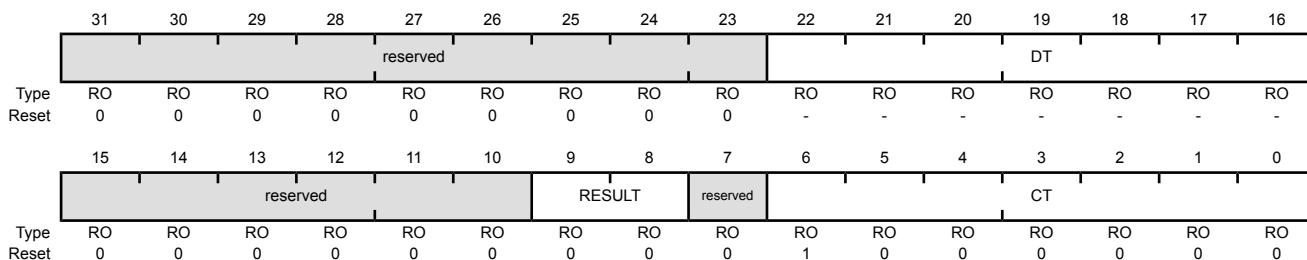
Bit/Field	Name	Type	Reset	Description
6:0	UT	R/W	0x0	User Trim Value User trim value that can be loaded into the PIOSC. Refer to “Main PLL Frequency Configuration” on page 209 for more information on calibrating the PIOSC.

Register 14: Precision Internal Oscillator Statistics (PIOSCSTAT), offset 0x154

This register provides the user information on the PIOSC calibration. Note that a 32.768-kHz oscillator must be used as the Hibernation module clock source for the user to be able to calibrate the PIOSC.

Precision Internal Oscillator Statistics (PIOSCSTAT)

Base 0x400F.E000
Offset 0x154
Type RO, reset 0x0000.0040



Bit/Field	Name	Type	Reset	Description
31:23	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22:16	DT	RO	-	Default Trim Value This field contains the default trim value. This value is loaded into the PIOSC after every full power-up.
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.
9:8	RESULT	RO	0	Calibration Result Value Description 0x0 Calibration has not been attempted. 0x1 The last calibration operation completed to meet 1% accuracy. 0x2 The last calibration operation failed to meet 1% accuracy. 0x3 Reserved
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	CT	RO	0x40	Calibration Trim Value This field contains the trim value from the last calibration operation. After factory calibration CT and DT are the same.

Register 15: I²S MCLK Configuration (I2SMCLKCFG), offset 0x170

This register configures the receive and transmit fractional clock dividers for the for the I²S master transmit and receive clocks (I2S0TXMCLK and I2S0RXMCLK) . Varying the integer and fractional inputs for the clocks allows greater accuracy in hitting the target I²S clock frequencies. Refer to “Clock Control” on page 772 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
Type	R/W	RO	R/W													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	R/W	RO	R/W													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description
31	RXEN	R/W	0	RX Clock Enable
				Value Description
			1	The I ² S receive clock generator is enabled.
			0	The I ² S receive clock generator is disabled. If the RXSLV bit in the I ² S Module Configuration (I2SCFG) register is set, then the I2S0RXMCLK must be externally generated.
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.
29:20	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
				Value Description
			1	The I ² S transmit clock generator is enabled.
			0	The I ² S transmit clock generator is disabled. If the TXSLV bit in the I ² S Module Configuration (I2SCFG) register is set, then the I2S0TXMCLK must be externally generated.
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.

Bit/Field	Name	Type	Reset	Description
13:4	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

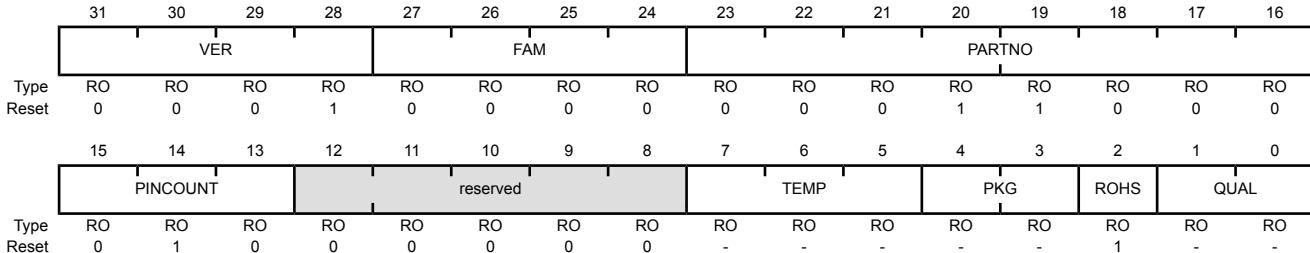
This register identifies the device family, part number, temperature range, and package type.

Device Identification 1 (DID1)

Base 0x400F.E000

Offset 0x004

Type RO, reset -



Bit/Field	Name	Type	Reset	Description
31:28	VER	RO	0x1	DID1 Version This field defines the DID1 register format version. The version number is numeric. The value of the VER field is encoded as follows (all other encodings are reserved): Value Description 0x1 Second version of the DID1 register format.
27:24	FAM	RO	0x0	Family This field provides the family identification of the device within the Luminary Micro product portfolio. The value is encoded as follows (all other encodings are reserved): Value Description 0x0 Stellaris family of microcontrollers, that is, all devices with external part numbers starting with LM3S.
23:16	PARTNO	RO	0x18	Part Number This field provides the part number of the device within the family. The value is encoded as follows (all other encodings are reserved): Value Description 0x18 LM3S9L97
15:13	PINCOUNT	RO	0x2	Package Pin Count This field specifies the number of pins on the device package. The value is encoded as follows (all other encodings are reserved): Value Description 0x2 100-pin package
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.

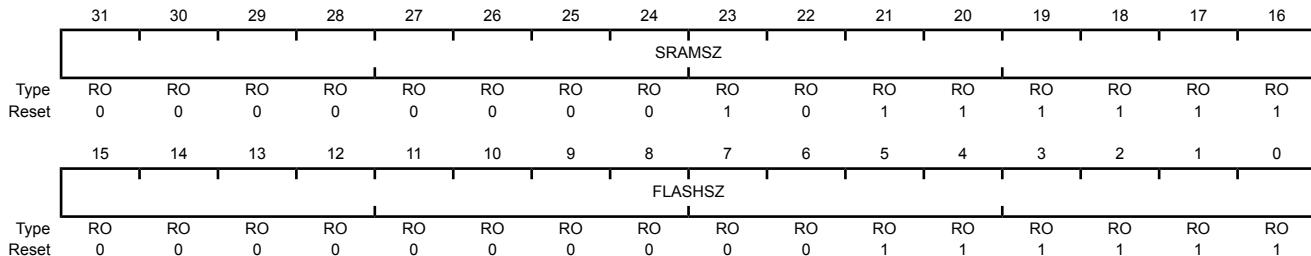
Bit/Field	Name	Type	Reset	Description								
7:5	TEMP	RO	-	<p>Temperature Range</p> <p>This field specifies the temperature rating of the device. The value is encoded as follows (all other encodings are reserved):</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Commercial temperature range (0°C to 70°C)</td></tr> <tr> <td>0x1</td><td>Industrial temperature range (-40°C to 85°C)</td></tr> <tr> <td>0x2</td><td>Extended temperature range (-40°C to 105°C)</td></tr> </tbody> </table>	Value	Description	0x0	Commercial temperature range (0°C to 70°C)	0x1	Industrial temperature range (-40°C to 85°C)	0x2	Extended temperature range (-40°C to 105°C)
Value	Description											
0x0	Commercial temperature range (0°C to 70°C)											
0x1	Industrial temperature range (-40°C to 85°C)											
0x2	Extended temperature range (-40°C to 105°C)											
4:3	PKG	RO	-	<p>Package Type</p> <p>This field specifies the package type. The value is encoded as follows (all other encodings are reserved):</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>SOIC package</td></tr> <tr> <td>0x1</td><td>LQFP package</td></tr> <tr> <td>0x2</td><td>BGA package</td></tr> </tbody> </table>	Value	Description	0x0	SOIC package	0x1	LQFP package	0x2	BGA package
Value	Description											
0x0	SOIC package											
0x1	LQFP package											
0x2	BGA package											
2	ROHS	RO	1	<p>RoHS-Compliance</p> <p>This bit specifies whether the device is RoHS-compliant. A 1 indicates the part is RoHS-compliant.</p>								
1:0	QUAL	RO	-	<p>Qualification Status</p> <p>This field specifies the qualification status of the device. The value is encoded as follows (all other encodings are reserved):</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Engineering Sample (unqualified)</td></tr> <tr> <td>0x1</td><td>Pilot Production (unqualified)</td></tr> <tr> <td>0x2</td><td>Fully Qualified</td></tr> </tbody> </table>	Value	Description	0x0	Engineering Sample (unqualified)	0x1	Pilot Production (unqualified)	0x2	Fully Qualified
Value	Description											
0x0	Engineering Sample (unqualified)											
0x1	Pilot Production (unqualified)											
0x2	Fully Qualified											

Register 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 0x00BF.003F



Bit/Field Name Type Reset Description

31:16 SRAMSZ RO 0x00BF SRAM Size
Indicates the size of the on-chip SRAM memory.

Value Description
0x00BF 48 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
Type	RO	ADC1	ADC0													
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
Type	RO	SWD	JTAG													
Reset	-	-	-	-	1	1	1	1	1	1	1	1	1	1	RO	RO

MINSYSDIV	MAXADC1SPD	MAXADC0SPD	MPU	HIB	TEMPSNS	PLL	WDTO	SWO	SWD	JTAG
-----------	------------	------------	-----	-----	---------	-----	------	-----	-----	------

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 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> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x1</td><td>Specifies an 80-MHz CPU clock with a PLL divider of 2.5.</td></tr> <tr> <td>0x2</td><td>Specifies a 66.67-MHz CPU clock with a PLL divider of 3.</td></tr> <tr> <td>0x3</td><td>Specifies a 50-MHz CPU clock with a PLL divider of 4.</td></tr> <tr> <td>0x7</td><td>Specifies a 25-MHz clock with a PLL divider of 8.</td></tr> <tr> <td>0x9</td><td>Specifies a 20-MHz clock with a PLL divider of 10.</td></tr> </tbody> </table>	Value	Description	0x1	Specifies an 80-MHz CPU clock with a PLL divider of 2.5.	0x2	Specifies a 66.67-MHz CPU clock with a PLL divider of 3.	0x3	Specifies a 50-MHz CPU clock with a PLL divider of 4.	0x7	Specifies a 25-MHz clock with a PLL divider of 8.	0x9	Specifies a 20-MHz clock with a PLL divider of 10.
Value	Description															
0x1	Specifies an 80-MHz CPU clock with a PLL divider of 2.5.															
0x2	Specifies a 66.67-MHz CPU clock with a PLL divider of 3.															
0x3	Specifies a 50-MHz CPU clock with a PLL divider of 4.															
0x7	Specifies a 25-MHz clock with a PLL divider of 8.															
0x9	Specifies a 20-MHz clock with a PLL divider of 10.															
11:10	MAXADC1SPD	RO	0x3	<p>Max ADC1 Speed This field indicates the maximum rate at which the ADC samples data.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>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	RO	0x3	<p>Max ADC0 Speed This field indicates the maximum rate at which the ADC samples data.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x3</td><td>1M samples/second</td></tr> </tbody> </table>	Value	Description	0x3	1M samples/second								
Value	Description															
0x3	1M samples/second															
7	MPU	RO	1	MPU Present When set, indicates that the Cortex-M3 Memory Protection Unit (MPU) module is present. See the "Cortex-M3 Peripherals" chapter in the Stellaris Data Sheet for details on the MPU.												
6	HIB	RO	1	Hibernation Module Present When set, indicates that the Hibernation module is present.												
5	TEMPSNS	RO	1	Temp Sensor Present When set, indicates that the on-chip temperature sensor is present.												
4	PLL	RO	1	PLL Present When set, indicates that the on-chip Phase Locked Loop (PLL) is present.												
3	WDT0	RO	1	Watchdog Timer 0 Present When set, indicates that watchdog timer 0 is present.												
2	SWO	RO	1	SWO Trace Port Present When set, indicates that the Serial Wire Output (SWO) trace port is present.												
1	SWD	RO	1	SWD Present When set, indicates that the Serial Wire Debugger (SWD) is present.												
0	JTAG	RO	1	JTAG Present When set, indicates that the JTAG debugger interface is present.												

Register 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 0x130F.5337

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	RO	RO	RO	RO	RO	RO	COMP1	COMP0		reserved			TIMER3	TIMER2	TIMER1	TIMER0
Reset	0	0	0	1	0	0	1	1	0	0	0	0	1	1	1	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO	RO	RO	RO	RO	RO	QEI1	QEI0		reserved	SSI1	SSI0	reserved	UART2	UART1	UART0
Reset	0	1	0	1	0	0	1	1	0	0	0	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	I2S0	RO	1	I2S Module 0 Present When set, indicates that I2S module 0 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	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.
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.

Bit/Field	Name	Type	Reset	Description
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	QEI1	RO	1	QEI Module 1 Present When set, indicates that QEI module 1 is present.
8	QEI0	RO	1	QEI Module 0 Present When set, indicates that QEI 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.
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.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	32KHZ	reserved	CCP5	CCP4	CCP3	CCP2	CCP1	CCP0	ADC0AIN7	ADC0AIN6	ADC0AIN5	ADC0AIN4	ADC0AIN3	ADC0AIN2	ADC0AIN1	ADC0AIN0
Reset	RO 1	RO 0	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	PWMFAULT	reserved	C1O	C1PLUS	C1MINUS	C0O	C0PLUS	C0MINUS	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0		
Reset	RO 1	RO 0	RO 0	RO 0	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1

Bit/Field	Name	Type	Reset	Description
31	32KHZ	RO	1	32KHz Input Clock Available When set, indicates 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.
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.

Bit/Field	Name	Type	Reset	Description
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.
14:12	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	C1O	RO	1	C1o Pin Present When set, indicates that the analog comparator 1 output pin is present.
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	C0O	RO	1	C0o Pin Present When set, indicates that the analog comparator 0 output pin is present.
7	C0PLUS	RO	1	C0+ Pin Present When set, indicates that the analog comparator 0 (+) input pin is present.
6	C0MINUS	RO	1	C0- Pin Present When set, indicates that the analog comparator 0 (-) input pin is present.
5	PWM5	RO	1	PWM5 Pin Present When set, indicates that the PWM pin 5 is present.
4	PWM4	RO	1	PWM4 Pin Present When set, indicates that the PWM pin 4 is present.
3	PWM3	RO	1	PWM3 Pin Present When set, indicates that the PWM pin 3 is present.
2	PWM2	RO	1	PWM2 Pin Present When set, indicates that the PWM pin 2 is present.
1	PWM1	RO	1	PWM1 Pin Present When set, indicates that the PWM pin 1 is present.
0	PWM0	RO	1	PWM0 Pin Present When set, indicates that the PWM pin 0 is present.

Register 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 0x5104.F1FF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	reserved	EPHY0	reserved	EMAC0	reserved			E1588	reserved			PICAL			reserved	
Reset	RO 0	RO 1	RO 0	RO 1	RO 0	RO 0	RO 0	RO 1	RO 0	RO 0	RO 0	RO 0	RO 0	RO 1	RO 0	RO 0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	CCP7	CCP6	UDMA	ROM	reserved			GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Reset	RO 1	RO 1	RO 1	RO 1	RO 0	RO 0	RO 0	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPHY0	RO	1	Ethernet PHY Layer 0 Present When set, indicates that Ethernet PHY layer 0 is present.
29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	EMAC0	RO	1	Ethernet MAC Layer 0 Present When set, indicates that Ethernet MAC layer 0 is present.
27:25	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	E1588	RO	1	1588 Capable When set, indicates that Ethernet MAC layer 0 is 1588 capable.
23:19	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
18	PICAL	RO	1	PIOSC Calibrate When set, indicates that the PIOSC can be calibrated by software.
17:16	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.

Bit/Field	Name	Type	Reset	Description
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.
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 0x0F30.003F

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	RO	RO	RO	RO	PWMFAULT3	PWMFAULT2	PWMFAULT1	PWMFAULT0	RO	RO	PWMEFLT	PWMESYNC	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
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
	reserved															

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	PWMESYNC	RO	1	PWM Extended SYNC Active When set, indicates that the PWM Extended SYNC feature is active.
19:6	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	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.

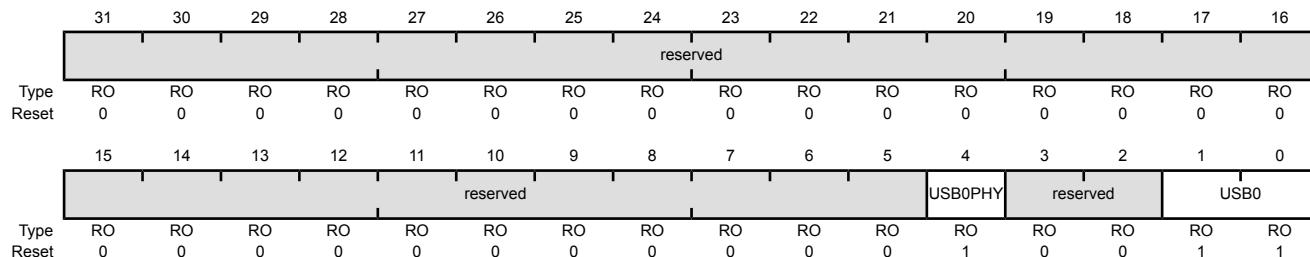
Bit/Field	Name	Type	Reset	Description
3	PWM3	RO	1	PWM3 Pin Present When set, indicates that the PWM pin 3 is present.
2	PWM2	RO	1	PWM2 Pin Present When set, indicates that the PWM pin 2 is present.
1	PWM1	RO	1	PWM1 Pin Present When set, indicates that the PWM pin 1 is present.
0	PWM0	RO	1	PWM0 Pin Present 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



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 secondary assignments. If the primary function is not available on this microcontroller, the secondary function becomes the primary function. If the secondary 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
Type	reserved	DMACH30	DMACH29	DMACH28	DMACH27	DMACH26	DMACH25	DMACH24	DMACH23	DMACH22	DMACH21	DMACH20	DMACH19	DMACH18	DMACH17	DMACH16
Reset	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	DMACH15	DMACH14	DMACH13	DMACH12	DMACH11	DMACH10	DMACH9	DMACH8	DMACH7	DMACH6	DMACH5	DMACH4	DMACH3	DMACH2	DMACH1	DMACH0
Reset	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1

Bit/Field	Name	Type	Reset	Description
31	reserved	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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary channel assignment of CAN module 2 receive.
21	DMACH21	RO	1	Timer1B / EPI0_WFIFO When set, indicates uDMA channel 21 is available and connected to Timer 1B. If the corresponding bit in the DMACHASGN register is set, the channel is connected instead to the secondary channel assignment of EPI module write FIFO (WRIFO).
20	DMACH20	RO	1	Timer1A / EPI0_NBRIFO When set, indicates uDMA channel 20 is available and connected to Timer 1A. If the corresponding bit in the DMACHASGN register is set, the channel is connected instead to the secondary channel assignment of EPI module 0 non-blocking read FIFO (NBRIFO).
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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary 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.
16	DMACH16	RO	1	ADC0_SS2 When set, indicates uDMA channel 16 is available and connected to ADC module 0 Sample Sequencer 2.

Bit/Field	Name	Type	Reset	Description
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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary channel assignment of UART module 2 receive.
11	DMACH11	RO	1	SSI0_TX / SSI1_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 DMACHASGN register is set, the channel is connected instead to the secondary channel assignment of SSI module 1 transmit.
10	DMACH10	RO	1	SSI0_RX / SSI1_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 DMACHASGN register is set, the channel is connected instead to the secondary channel assignment of SSI module 1 receive.
9	DMACH9	RO	1	UART0_TX / UART1_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 DMACHASGN register is set, the channel is connected instead to the secondary channel assignment of UART module 1 transmit.
8	DMACH8	RO	1	UART0_RX / UART1_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 DMACHASGN register is set, the channel is connected instead to the secondary channel assignment of UART module 1 receive.
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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary channel assignment of Timer 2A.

Bit/Field	Name	Type	Reset	Description
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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary 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 DMACHASGN register is set, the channel is connected instead to the secondary 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
Type	ADC1AIN15	ADC1AIN14	ADC1AIN13	ADC1AIN12	ADC1AIN11	ADC1AIN10	ADC1AIN9	ADC1AIN8	ADC1AIN7	ADC1AIN6	ADC1AIN5	ADC1AIN4	ADC1AIN3	ADC1AIN2	ADC1AIN1	ADC1AIN0
Reset	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1					
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	ADC0AIN15	ADC0AIN14	ADC0AIN13	ADC0AIN12	ADC0AIN11	ADC0AIN10	ADC0AIN9	ADC0AIN8	ADC0AIN7	ADC0AIN6	ADC0AIN5	ADC0AIN4	ADC0AIN3	ADC0AIN2	ADC0AIN1	ADC0AIN0
Reset	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1	RO 1					

Bit/Field	Name	Type	Reset	Description
31	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.
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.

Bit/Field	Name	Type	Reset	Description
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.
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	1	1	1	1	1	1	1	1
Reset	0	0	0	0	0	0	0	0	0	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.
6	ADC0DC6	RO	1	ADC0 DC6 Present When set, indicates that ADC module 0 Digital Comparator 6 is present.

Bit/Field	Name	Type	Reset	Description
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															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	FWB														
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 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. **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
Type	reserved			WDT1	reserved		CAN1	CANO	reserved			PWM	reserved		ADC1	ADC0
Reset	RO 0	RO 0	RO 0	R/W 0	RO 0	RO 0	R/W 0	R/W 0	RO 0	RO 0	RO 0	R/W 0	RO 0	RO 0	R/W 0	R/W 0
Type	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	reserved			MAXADC1SPD		MAXADC0SPD		reserved	HIB	reserved		WDT0	reserved			
Type	RO 0	RO 0	RO 0	RO 0	R/W 0	R/W 0	R/W 0	R/W 0	RO 0	R/W 1	RO 0	RO 0	R/W 0	RO 0	RO 0	RO 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
24	CANO	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 unclocked and disabled. If the module is unclocked, 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	PWM Clock Gating Control This bit controls the clock gating for the PWM module. 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.										
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	ADC1 Clock Gating Control This bit controls the clock gating for SAR ADC 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.										
16	ADC0	R/W	0	ADC0 Clock Gating Control This bit controls the clock gating for ADC 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.										
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	ADC1 Sample Speed 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): <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>1M samples/second</td> </tr> <tr> <td>0x2</td> <td>500K samples/second</td> </tr> <tr> <td>0x1</td> <td>250K samples/second</td> </tr> <tr> <td>0x0</td> <td>125K samples/second</td> </tr> </tbody> </table>	Value	Description	0x3	1M samples/second	0x2	500K samples/second	0x1	250K samples/second	0x0	125K samples/second
Value	Description													
0x3	1M samples/second													
0x2	500K samples/second													
0x1	250K samples/second													
0x0	125K samples/second													
9:8	MAXADC0SPD	R/W	0	ADC0 Sample Speed 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): <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x3</td> <td>1M samples/second</td> </tr> <tr> <td>0x2</td> <td>500K samples/second</td> </tr> <tr> <td>0x1</td> <td>250K samples/second</td> </tr> <tr> <td>0x0</td> <td>125K samples/second</td> </tr> </tbody> </table>	Value	Description	0x3	1M samples/second	0x2	500K samples/second	0x1	250K samples/second	0x0	125K samples/second
Value	Description													
0x3	1M samples/second													
0x2	500K samples/second													
0x1	250K samples/second													
0x0	125K samples/second													

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	HIB	R/W	1	HIB Clock Gating Control This bit controls the clock gating for the Hibernation module. 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.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	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 unclocked and disabled. If the module is unclocked, 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 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. **RCGC0** is the clock configuration register for running operation, **SCGC0** for Sleep operation, and **DCGC0** for Deep-Sleep operation. Setting the ACG bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

Sleep Mode Clock Gating Control Register 0 (SCGC0)

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

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

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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
24	CANO	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 unclocked and disabled. If the module is unclocked, 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 This bit controls the clock gating for the PWM module. 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>										
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 This bit controls the clock gating for ADC 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.</p>										
16	ADC0	R/W	0	<p>ADC0 Clock Gating Control This bit controls the clock gating for ADC 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>										
15:12	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 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> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x3</td><td>1M samples/second</td></tr> <tr> <td>0x2</td><td>500K samples/second</td></tr> <tr> <td>0x1</td><td>250K samples/second</td></tr> <tr> <td>0x0</td><td>125K samples/second</td></tr> </tbody> </table>	Value	Description	0x3	1M samples/second	0x2	500K samples/second	0x1	250K samples/second	0x0	125K samples/second
Value	Description													
0x3	1M samples/second													
0x2	500K samples/second													
0x1	250K samples/second													
0x0	125K samples/second													
9:8	MAXADC0SPD	R/W	0	<p>ADC0 Sample Speed 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> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x3</td><td>1M samples/second</td></tr> <tr> <td>0x2</td><td>500K samples/second</td></tr> <tr> <td>0x1</td><td>250K samples/second</td></tr> <tr> <td>0x0</td><td>125K samples/second</td></tr> </tbody> </table>	Value	Description	0x3	1M samples/second	0x2	500K samples/second	0x1	250K samples/second	0x0	125K samples/second
Value	Description													
0x3	1M samples/second													
0x2	500K samples/second													
0x1	250K samples/second													
0x0	125K samples/second													

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	HIB	R/W	1	HIB Clock Gating Control This bit controls the clock gating for the Hibernation module. 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.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	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 unclocked and disabled. If the module is unclocked, 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 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. **RCGC0** is the clock configuration register for running operation, **SCGC0** for Sleep operation, and **DCGC0** for Deep-Sleep operation. Setting the ACG bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

Deep Sleep Mode Clock Gating Control Register 0 (DCGC0)

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

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

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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
24	CANO	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 unclocked and disabled. If the module is unclocked, 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	PWM Clock Gating Control This bit controls the clock gating for the PWM module. 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.
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	ADC1 Clock Gating Control This bit controls the clock gating for ADC 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.
16	ADC0	R/W	0	ADC0 Clock Gating Control This bit controls the clock gating for ADC 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.
15:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	HIB	R/W	1	HIB Clock Gating Control This bit controls the clock gating for the Hibernation module. 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.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	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 unclocked and disabled. If the module is unclocked, 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
Type	reserved			I2S0	reserved		COMP1	COMP0	reserved				TIMER3	TIMER2	TIMER1	TIMER0
Reset	RO 0	RO 0	RO 0	R/W 0	RO 0	RO 0	R/W 0	R/W 0	RO 0	RO 0	RO 0	RO 0	R/W 0	R/W 0	R/W 0	R/W 0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	reserved	I2C1	reserved	I2C0	reserved		QEI1	QEI0	reserved		SSI1	SSI0	reserved	UART2	UART1	UART0
Reset	RO 0	R/W 0	RO 0	R/W 0	RO 0	RO 0	R/W 0	R/W 0	RO 0	RO 0	R/W 0	R/W 0	RO 0	R/W 0	R/W 0	R/W 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	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.
27:26	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	COMP1	R/W	0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 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.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 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.

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 Clock Gating Control 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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the 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.
17	TIMER1	R/W	0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the 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.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the 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.
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 Clock Gating Control This bit controls the clock gating for I2C 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.
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 unclocked and disabled. If the module is unclocked, 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	QEI1	R/W	0	QEI1 Clock Gating Control This bit controls the clock gating for QEI 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.

Bit/Field	Name	Type	Reset	Description
8	QEI0	R/W	0	QEI0 Clock Gating Control This bit controls the clock gating for QEI 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.
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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	UART1	R/W	0	UART1 Clock Gating Control This bit controls the clock gating for UART module 1. If set, the 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 **DGCG1** for Deep-Sleep operation. Setting the ACG bit in the **Run-Mode Clock Configuration (RCC)** register specifies that the system uses sleep modes.

Sleep Mode Clock Gating Control Register 1 (SCGC1)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	reserved	I2S0	reserved	COMP1	COMP0	reserved	reserved	reserved	reserved	TIMER3	TIMER2	TIMER1	TIMER0			
Reset	RO 0	RO 0	RO 0	R/W 0	RO 0	RO 0	R/W 0	R/W 0	RO 0	RO 0	RO 0	RO 0	R/W 0	R/W 0	R/W 0	R/W 0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	reserved	I2C1	reserved	I2C0	reserved	QEI1	QEI0	reserved	SSI1	SSI0	reserved	UART2	UART1	UART0		
Reset	RO 0	R/W 0	RO 0	R/W 0	RO 0	RO 0	R/W 0	R/W 0	RO 0	RO 0	R/W 0	RO 0	R/W 0	R/W 0	R/W 0	R/W 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	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.
27:26	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	COMP1	R/W	0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 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.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 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.

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 Clock Gating Control 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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the 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.
17	TIMER1	R/W	0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the 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.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the 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.
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 Clock Gating Control This bit controls the clock gating for I2C 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.
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 unclocked and disabled. If the module is unclocked, 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	QEI1	R/W	0	QEI1 Clock Gating Control This bit controls the clock gating for QEI 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.

Bit/Field	Name	Type	Reset	Description
8	QEI0	R/W	0	QEI0 Clock Gating Control This bit controls the clock gating for QEI 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.
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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	UART1	R/W	0	UART1 Clock Gating Control This bit controls the clock gating for UART module 1. If set, the 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
Type	RO	RO	RO	R/W	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
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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
31:29	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.
27:26	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	COMP1	R/W	0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 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.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 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.

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 Clock Gating Control 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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the 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.
17	TIMER1	R/W	0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the 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.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the 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.
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 Clock Gating Control This bit controls the clock gating for I2C 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.
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 unclocked and disabled. If the module is unclocked, 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	QEI1	R/W	0	QEI1 Clock Gating Control This bit controls the clock gating for QEI 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.

Bit/Field	Name	Type	Reset	Description
8	QEI0	R/W	0	QEI0 Clock Gating Control This bit controls the clock gating for QEI 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.
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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	UART1	R/W	0	UART1 Clock Gating Control This bit controls the clock gating for UART module 1. If set, the 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 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.

Run Mode Clock Gating Control Register 2 (RCGC2)

Base 0x400F.E000
Offset 0x108
Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Type	reserved	EPHY0	reserved	EMAC0	reserved												USBO
Type	RO	R/W	RO	R/W	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Type	reserved		UDMA	reserved				GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	
Type	RO	RO	R/W	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	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPHY0	R/W	0	PHY0 Clock Gating Control This bit controls the clock gating for Ethernet PHY layer 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	EMAC0	R/W	0	MAC0 Clock Gating Control This bit controls the clock gating for Ethernet MAC layer 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.
27:17	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.

Bit/Field	Name	Type	Reset	Description
16	USB0	R/W	0	USB0 Clock Gating Control 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.
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	Micro-DMA Clock Gating Control 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.
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	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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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	
Type	reserved	EPHY0	reserved	EMAC0	reserved												USBO
Reset	RO 0	R/W 0	RO 0	R/W 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	R/W 0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Type	reserved	UDMA		reserved			GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA		
Reset	RO 0	RO 0	R/W 0	RO 0	RO 0	RO 0	RO 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPHY0	R/W	0	PHY0 Clock Gating Control This bit controls the clock gating for Ethernet PHY layer 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	EMAC0	R/W	0	MAC0 Clock Gating Control This bit controls the clock gating for Ethernet MAC layer 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.
27:17	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.

Bit/Field	Name	Type	Reset	Description
16	USB0	R/W	0	USB0 Clock Gating Control 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.
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	Micro-DMA Clock Gating Control 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.
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	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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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
Type	reserved	EPHY0	reserved	EMAC0						reserved						USBO
Reset	RO 0	R/W 0	RO 0	R/W 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	R/W 0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	reserved	UDMA		reserved			GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	
Reset	RO 0	RO 0	R/W 0	RO 0	RO 0	RO 0	RO 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPHY0	R/W	0	PHY0 Clock Gating Control This bit controls the clock gating for Ethernet PHY layer 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	EMAC0	R/W	0	MAC0 Clock Gating Control This bit controls the clock gating for Ethernet MAC layer 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.
27:17	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.

Bit/Field	Name	Type	Reset	Description
16	USB0	R/W	0	USB0 Clock Gating Control 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.
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	Micro-DMA Clock Gating Control 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.
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	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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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 unclocked and disabled. If the module is unclocked, 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
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
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	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.
15:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	HIB	R/W	0	HIB Reset Control When this bit is set, the Hibernation 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.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	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
Type	RO	RO	RO	R/W	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
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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
31:29	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.
27:26	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	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.
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.

Bit/Field	Name	Type	Reset	Description
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	QEI1	R/W	0	QEI1 Reset Control When this bit is set, QEI 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	QEI0	R/W	0	QEI0 Reset Control When this bit is set, QEI 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.
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.

Bit/Field	Name	Type	Reset	Description
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
Type	reserved	EPHY0	reserved	EMAC0						reserved						USBO
Reset	RO 0	R/W 0	RO 0	R/W 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	R/W 0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	reserved	UDMA		reserved				GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Reset	RO 0	RO 0	R/W 0	RO 0	RO 0	RO 0	RO 0	R/W 0								

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPHY0	R/W	0	PHY0 Reset Control When this bit is set, Ethernet PHY layer 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	EMAC0	R/W	0	MAC0 Reset Control When this bit is set, Ethernet MAC layer 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.
27:17	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	USBO	R/W	0	USB0 Reset Control 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.
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	Micro-DMA Reset Control 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.
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 Reset Control 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.
7	GPIOH	R/W	0	Port H Reset Control 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.
6	GPIOG	R/W	0	Port G Reset Control 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.
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.

6 Hibernation Module

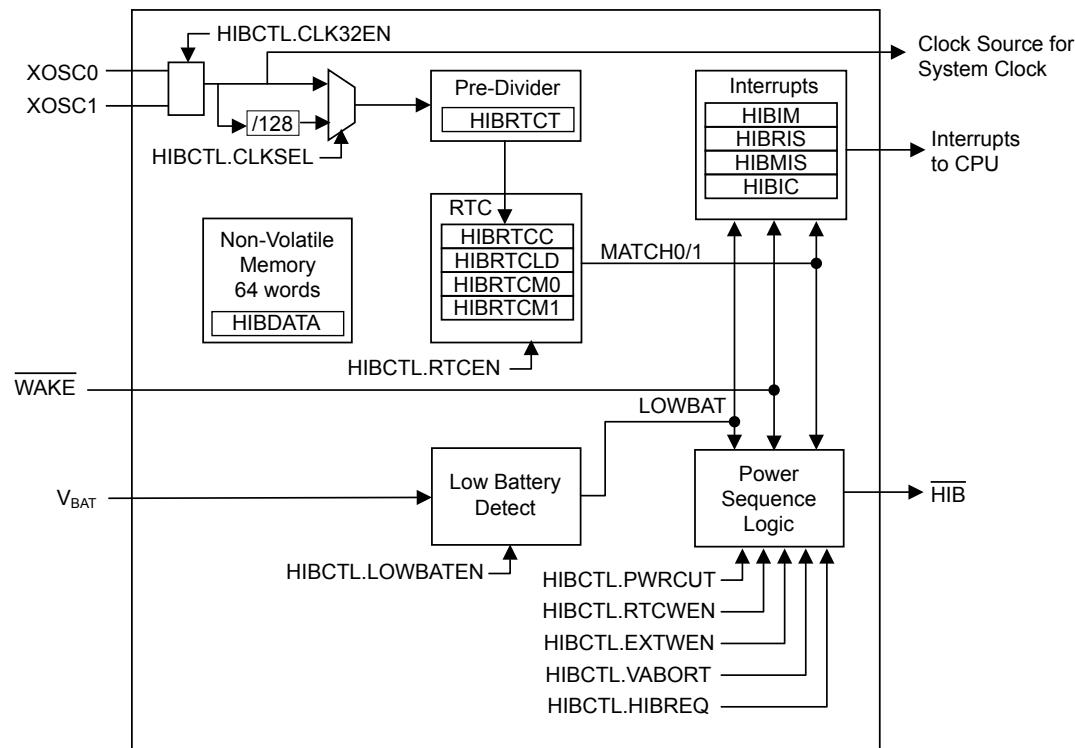
The Hibernation Module manages removal and restoration of power to provide a means for reducing power consumption. When the processor and peripherals are idle, power can be completely removed with only the Hibernation module remaining powered. Power can be restored based on an external signal or at a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

The Hibernation module has the following features:

- 32-bit real-time counter (RTC)
 - Two 32-bit RTC match registers for timed wake-up and interrupt generation
 - RTC predivider trim for making fine adjustments to the clock rate
- Two mechanisms for power control
 - System power control using discrete external regulator
 - On-chip power control using internal switches under register control
- Dedicated pin for waking using an external signal
- RTC operational and hibernation memory valid as long as V_{BAT} is valid
- Low-battery detection, signaling, and interrupt generation
- Clock source from a 32.768-kHz external oscillator or a 4.194304-MHz crystal; 32.768-kHz external oscillator can be used for main controller clock
- 64 32-bit words of non-volatile memory to save state during hibernation
- Programmable interrupts for RTC match, external wake, and low battery events

6.1 Block Diagram

Figure 6-1. Hibernation Module Block Diagram



6.2 Signal Description

Table 6-1 on page 302 and Table 6-2 on page 303 list the external signals of the Hibernation module and describe the function of each. These signals have dedicated functions and are not alternate functions for any GPIO signals.

Table 6-1. Signals for Hibernate (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
HIB	51	fixed	O	OD	An open-drain output that indicates the processor is in Hibernate mode.
V _{BAT}	55	fixed	-	Power	Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply.
WAKE	50	fixed	I	TTL	An external input that brings the processor out of Hibernate mode when asserted.
XOSC0	52	fixed	I	Analog	Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the HIBCTL register.

Table 6-1. Signals for Hibernate (100LQFP) (continued)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
XOSC1	53	fixed	O	Analog	Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source.

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

Table 6-2. Signals for Hibernate (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
HIB	M12	fixed	O	OD	An open-drain output that indicates the processor is in Hibernate mode.
VBAT	L12	fixed	-	Power	Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply.
WAKE	M10	fixed	I	TTL	An external input that brings the processor out of Hibernate mode when asserted.
XOSC0	K11	fixed	I	Analog	Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the HIBCTL register.
XOSC1	K12	fixed	O	Analog	Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source.

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

6.3 Functional Description

Important: The Hibernate module must have either the RTC function or the External Wake function enabled to ensure proper operation of the microcontroller. See “Initialization” on page 308.

The Hibernation module provides two mechanisms for power control:

- The first mechanism controls the power to the microcontroller with a control signal (**HIB**) that signals an external voltage regulator to turn on or off.
- The second mechanism uses internal switches to control power to the Cortex-M3 as well as to most analog and digital functions while retaining I/O pin power (VDD3ON mode).

The Hibernation module power source is determined dynamically. The supply voltage of the Hibernation module is the larger of the main voltage source (V_{DD}) or the battery/auxilliary voltage source (V_{BAT}). Care must be taken when using a single-ended clock source that the voltage amplitude of the 32.768-kHz Hibernation oscillator is less than V_{BAT} , otherwise, the Hibernation module draws power from the oscillator and not V_{BAT} . The Hibernation module also has an independent clock source to maintain a real-time clock (RTC) when the system clock is powered down. Once in hibernation, the module signals an external voltage regulator to turn the power back on when an external pin (**WAKE**) is asserted or when the internal RTC reaches a certain value. The Hibernation module can also detect when the battery voltage is low and optionally prevent hibernation when this occurs.

When waking from hibernation, the $\overline{\text{HIB}}$ signal is deasserted. The return of V_{DD} causes a POR to be executed. The time from when the $\overline{\text{WAKE}}$ signal is asserted to when code begins execution is equal to the wake-up time ($t_{\text{WAKE_TO_HIB}}$) plus the power-on reset time (T_{IRPOR}).

6.3.1 Register Access Timing

Because the Hibernation module has an independent clocking domain, certain registers must be written only with a timing gap between accesses. The delay time is $t_{\text{HIB_REG_ACCESS}}$, therefore software must guarantee that this delay is inserted between back-to-back writes to certain Hibernation registers or between a write followed by a read to those same registers. Software may make use of the **WRC** bit in the **Hibernation Control (HIBCTL)** register to ensure that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **HIBCTL** for **WRC=1** prior to accessing any affected register. The following registers are subject to this timing restriction:

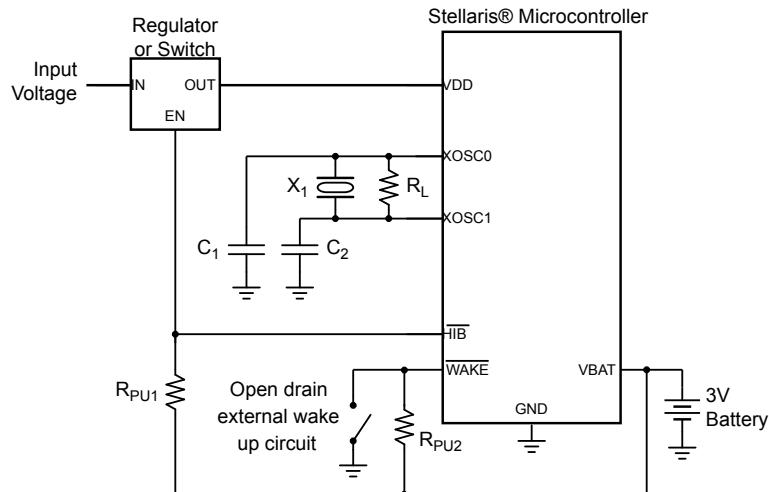
- **Hibernation RTC Counter (HIBRTCC)**
- **Hibernation RTC Match 0 (HIBRTCM0)**
- **Hibernation RTC Match 1 (HIBRTCM1)**
- **Hibernation RTC Load (HIBRTCLD)**
- **Hibernation RTC Trim (HIBRTCT)**
- **Hibernation Data (HIBDATA)**

Back-to-back reads from Hibernation module registers have no timing restrictions. Reads are performed at the full peripheral clock rate.

6.3.2 Hibernation Clock Source

In systems where the Hibernation module is used to put the microcontroller into hibernation, the module must be clocked by an external source that is independent from the main system clock, even if the RTC feature is not used. An external oscillator or crystal is used for this purpose. To use a crystal, a 4.194304-MHz crystal is connected to the xosc0 and xosc1 pins. This clock signal is divided by 128 internally to produce a 32.768-kHz Hibernation clock reference. Alternatively, a 32.768-kHz oscillator can be connected to the xosc0 pin, leaving xosc1 unconnected. Care must be taken that the voltage amplitude of the 32.768-kHz oscillator is less than V_{BAT} , otherwise, the Hibernation module may draw power from the oscillator and not V_{BAT} during hibernation. See Figure 6-2 on page 305 and Figure 6-3 on page 305. Note that these diagrams only show the connection to the Hibernation pins and not to the full system. See “Hibernation Module” on page 1241 for specific values.

The Hibernation clock source is enabled by setting the **CLK32EN** bit of the **HIBCTL** register. The type of clock source is selected by clearing the **CLKSEL** bit for a 4.194304-MHz crystal and setting the **CLKSEL** bit for a 32.768-kHz oscillator. If a crystal is used for the clock source, the software must leave a delay of $t_{\text{xosc_SETTLE}}$ after writing to the **CLK32EN** bit and before any other accesses to the Hibernation module registers. The delay allows the crystal to power up and stabilize. If an oscillator is used for the clock source, no delay is needed.

Figure 6-2. Using a Crystal as the Hibernation Clock Source

Note: X_1 = Crystal frequency is f_{XOSC_XTAL} .

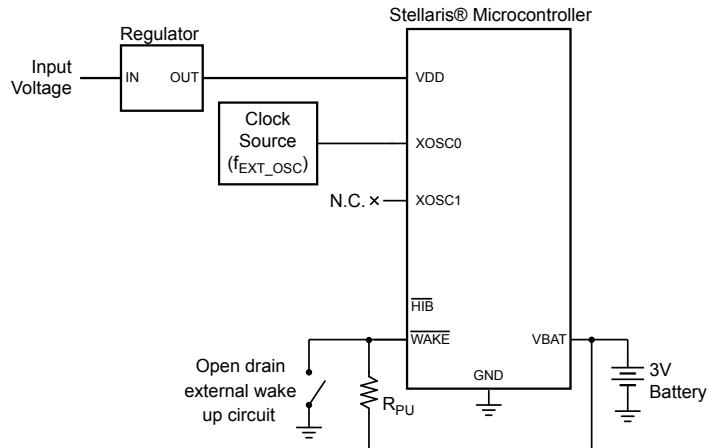
$C_{1,2}$ = Capacitor value derived from crystal vendor load capacitance specifications.

R_L = Load resistor is R_{XOSC_LOAD} .

R_{PU1} = Pull-up resistor 1 (value and voltage source (V_{BAT} or Input Voltage) determined by regulator or switch enable input characteristics).

R_{PU2} = Pull-up resistor 2 is 1 MΩ

See “Hibernation Module” on page 1241 for specific parameter values.

Figure 6-3. Using a Dedicated Oscillator as the Hibernation Clock Source with VDD3ON Mode

Note: R_{PU} = Pull-up resistor is 1 MΩ

If the application does not require the use of the Hibernation module, the XOSC0 and XOSC1 can remain unconnected. In this situation, the HIB bit in the **Run Mode Clock Gating Control Register 0 (RCGC0)** register must be cleared, disabling the system clock to the Hibernation module and Hibernation module registers are not accessible.

6.3.2.1 Special Considerations When Using a 4.194304-MHz Crystal

For some 4.194304-MHz crystals, the manufacturer-recommended crystal value may be outside of the capabilities of the hibernate module oscillator. If the crystal manufacturer's recommended load capacitance is used, the hibernate oscillator may fail to start.

For a parallel-resonant oscillator circuit, the total load capacitance C_L (as specified by the manufacturer) is calculated as follows:

$$C_L = (C_1 * C_2) / (C_1 + C_2) + C_S$$

The internal oscillator was designed for a typical C_1 and C_2 of 16 pF (C_1 and C_2 are specified as 12 pF minimum and 22 pF maximum). Using 2 pF for stray capacitance (C_S) and the typical value of 16 pF for C_1 and C_2 , the formula above shows that the selected crystal should have a C_L specification of about 10 pF. If the crystal has a C_L specification higher than 13 pF or lower than 8 pF, or if C_S is substantially different from 2 pF, then the oscillator frequency may be outside of the specified accuracy. The crystal manufacturer can provide this error information.

6.3.3 Battery Management

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

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

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

The Hibernation module can be configured to detect a low battery condition by setting the `LOWBATEN` bit of the **HIBCTL** register. In this configuration, the `LOWBAT` bit of the **Hibernation Raw Interrupt Status (HIBRIS)** register is set when the battery level is low. If the `VABORT` bit in the **HIBCTL** register is also set, then the module is prevented from entering Hibernation mode when a low battery is detected. The module can also be configured to generate an interrupt for the low-battery condition (see “Interrupts and Status” on page 308).

6.3.4 Real-Time Clock

The Hibernation module includes a 32-bit counter that increments once per second with the proper configuration (see “Hibernation Clock Source” on page 304). The 32.768-kHz clock signal, either directly from the 32.768-kHz oscillator or from the 4.194304-MHz crystal divided by 128, is fed into a predivider register that counts down the 32.768-kHz clock ticks to achieve a once per second clock rate for the RTC. A buffered version of the 32.768-kHz signal is available on the `RTCCLK` output. The rate can be adjusted to compensate for inaccuracies in the clock source by using the predivider trim register, **HIBRTCT**. This register has a nominal value of 0x7FFF, and is used for one second out of every 64 seconds to divide the input clock. This configuration allows the software to make fine corrections to the clock rate by adjusting the predivider trim register up or down from 0x7FFF. The predivider trim should be adjusted up from 0x7FFF in order to slow down the RTC rate and down from 0x7FFF in order to speed up the RTC rate.

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

The RTC must be enabled with the **RTCEN** bit of the **HIBCTL** register. The value of the RTC can be set at any time by writing to the **HIBRTCLD** register. The predivider trim can be adjusted by reading and writing the **HIBRTCT** register. The predivider uses this register once every 64 seconds to adjust the clock rate. The two match registers can be set by writing to the **HIBRTCM0** and **HIBRTCM1** registers. The RTC can be configured to generate interrupts by using the interrupt registers (see “Interrupts and Status” on page 308).

6.3.5 Non-Volatile Memory

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

6.3.6 Power Control Using **HIB**

Important: The Hibernation Module requires special system implementation considerations when using **HIB** to control power, as it is intended to power-down all other sections of the microcontroller. All system signals and power supplies that connect to the chip must be driven to 0 V_{DC} or powered down with the same regulator controlled by **HIB**. See “Hibernation Module” on page 1241 for more details.

The Hibernation module controls power to the microcontroller through the use of the **HIB** pin which is intended to be connected to the enable signal of the external regulator(s) providing 3.3 V to the microcontroller and other circuits. When the **HIB** signal is asserted by the Hibernation module, the external regulator is turned off and no longer powers the microcontroller and any parts of the system that are powered by the regulator. The Hibernation module remains powered from the V_{BAT} supply (which could be a battery or an auxiliary power source) until a Wake event. Power to the microcontroller is restored by deasserting the **HIB** signal, which causes the external regulator to turn power back on to the chip.

6.3.7 Power Control Using VDD3ON Mode

The Hibernation module may also be configured to cut power to all internal modules. While in this state, all pins are configured as inputs. In the VDD3ON mode, the regulator should maintain 3.3 V power to the microcontroller during Hibernate. This power control mode is enabled by setting the **VDD3ON** bit in **HIBCTL**.

6.3.8 Initiating Hibernate

Prior to initiating hibernation, a wake-up condition must be configured, either from the external **WAKE** pin, or by using an RTC match. Hibernation mode is initiated when the **HIBREQ** bit of the **HIBCTL** register is set. If a Flash memory write operation is in progress, an interlock feature holds off the transition into Hibernation mode until the write has completed.

The Hibernation module is configured to wake from the external **WAKE** pin by setting the **PINWEN** bit of the **HIBCTL** register. It is configured to wake from RTC match by setting the **RTCWEN** bit. Either one or both of these bits must be set prior to going into hibernation. Note that the **WAKE** pin uses the Hibernation module's internal power supply as the logic 1 reference.

Upon either external wake-up or RTC match, the Hibernation module delays coming out of hibernation until V_{DD} is above the minimum specified voltage, see Table 26-2 on page 1232.

When the Hibernation module wakes, the microcontroller performs a normal power-on reset. Software can detect that the power-on was due to a wake from hibernation by examining the raw interrupt status register (see “Interrupts and Status” on page 308) and by looking for state data in the non-volatile memory (see “Non-Volatile Memory” on page 307).

6.3.9 Interrupts and Status

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

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

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

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

6.4 Initialization and Configuration

The Hibernation module has several different configurations. The following sections show the recommended programming sequence for various scenarios. The examples below assume that a 32.768-kHz oscillator is used, and thus always set the **CLKSEL** bit of the **HIBCTL** register. If a 4.194304-MHz crystal is used instead, then the **CLKSEL** bit remains cleared. Because the Hibernation module runs at 32.768 kHz and is asynchronous to the rest of the microcontroller, which is run off the system clock, software must allow a delay of $t_{HIB_REG_ACCESS}$ after writes to certain registers (see “Register Access Timing” on page 304). The registers that require a delay are listed in a note in “Register Map” on page 311 as well as in each register description.

6.4.1 Initialization

The Hibernation module comes out of reset with the system clock enabled to the module, but if the system clock to the module has been disabled, then it must be re-enabled, even if the RTC feature is not used. See page 268.

If a 4.194304-MHz crystal is used as the Hibernation module clock source, perform the following steps:

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

If a 32.678-kHz oscillator is used as the Hibernation module clock source, then perform the following steps:

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

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

Table 6-3 on page 309 illustrates how the clocks function with various bit setting both in normal operation and in hibernation.

Table 6-3. Hibernation Module Clock Operation

CLK32EN	PINWEN	RTCWEN	CLKSEL	RTCEN	Result Normal Operation	Result Hibernation
0	X	X	X	X	Hibernation module disabled	Hibernation module disabled
1	0	0	0	1	RTC match capability enabled. Module clocked from 4.184304-MHz crystal.	No hibernation
1	0	0	1	1	RTC match capability enabled. Module clocked from 32.768-kHz oscillator.	No hibernation
1	0	1	X	1	Module clocked from selected source	RTC match for wake-up event
1	1	0	X	0	Module clocked from selected source	Clock is powered down during hibernation and powered up again on external wake-up event.
1	1	0	X	1	Module clocked from selected source	Clock is powered up during hibernation for RTC. Wake up on external event.
1	1	1	X	1	Module clocked from selected source	RTC match or external wake-up event, whichever occurs first.

6.4.2 RTC Match Functionality (No Hibernation)

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

1. Write the required RTC match value to one of the **HIBRTCMn** registers at offset 0x004 or 0x008.
2. Write the required RTC load value to the **HIBRTCLD** register at offset 0x00C.
3. Set the required RTC match interrupt mask in the RTCALT0 and RTCALT1 bits (bits 1:0) in the **HIBIM** register at offset 0x014.
4. Write 0x0000.0041 to the **HIBCTL** register at offset 0x010 to enable the RTC to begin counting.

6.4.3 RTC Match/Wake-Up from Hibernation

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

1. Write the required RTC match value to the **HIBRTCMn** registers at offset 0x004 or 0x008.

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

6.4.4 External Wake-Up from Hibernation

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

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

Note that in this mode, if the RTC is disabled, then the Hibernation clock source is powered down during Hibernation mode and is powered up again on the external wake event to save power during hibernation. If the RTC is enabled before hibernation, it will continue to operate during hibernation.

6.4.5 RTC or External Wake-Up from Hibernation

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

6.4.6 Register Reset

The Hibernation module handles resets according to the following conditions:

- Cold Reset

When the hibernation module has no externally applied voltage and detects a change to either V_{DD} or V_{BAT} , it resets all hibernation module registers to the value in Table 6-4 on page 311.

- Reset During Hibernation Module Disable

When the module has either not been enabled or has been disabled by software, the reset is passed through to the Hibernation module circuitry, and the internal state of the module is reset. Non-volatile memory contents are not reset to zero and contents after reset are indeterminate.

- Reset While Hibernation Module is in Hibernation Mode

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

- Reset While Hibernation Module is in Normal Mode

While in normal mode (not hibernating), any reset is suppressed if either the RTCEN or the PINWEN bit is set in the **HIBCTL** register, and the content/state of the control and data registers is unaffected.

Software must initialize any control or data registers in this condition. Therefore, software is the only mechanism to set or clear the CLK32EN bit and real-time clock operation, or to clear contents of the data memory. The only state that must be cleared by a reset operation while not in Hibernation mode is any state that prevents software from managing the interface.

Note: If V_{DD} drops below operational range while in normal mode (not hibernating), all hibernation module registers are reset to the value in Table 6-4 on page 311, regardless of whether the proper voltage is applied to V_{BAT} .

6.5 Register Map

Table 6-4 on page 311 lists the Hibernation registers. All addresses given are relative to the Hibernation Module base address at 0x400F.C000. Note that the system clock to the Hibernation module must be enabled before the registers can be programmed (see page 268).

Note: **HIBRTC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See “Register Access Timing” on page 304.

Important: Reset values apply only to a cold reset. Once configured, the Hibernate module ignores any system reset, other than power on reset, as long as V_{BAT} is present.

Table 6-4. Hibernation Module Register Map

Offset	Name	Type	Reset	Description	See page
0x000	HIBRTC	RO	0x0000.0000	Hibernation RTC Counter	312
0x004	HIBRTCM0	R/W	0xFFFF.FFFF	Hibernation RTC Match 0	313
0x008	HIBRTCM1	R/W	0xFFFF.FFFF	Hibernation RTC Match 1	314
0x00C	HIBRTCLD	R/W	0xFFFF.FFFF	Hibernation RTC Load	315
0x010	HIBCTL	R/W	0x8000.0000	Hibernation Control	316
0x014	HIBIM	R/W	0x0000.0000	Hibernation Interrupt Mask	319
0x018	HIBRIS	RO	0x0000.0000	Hibernation Raw Interrupt Status	321
0x01C	HIBMIS	RO	0x0000.0000	Hibernation Masked Interrupt Status	323
0x020	HIBIC	R/W1C	0x0000.0000	Hibernation Interrupt Clear	325
0x024	HIBRTCT	R/W	0x0000.7FFF	Hibernation RTC Trim	326
0x030-0x12C	HIBDATA	R/W	-	Hibernation Data	327

6.6 Register Descriptions

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

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

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

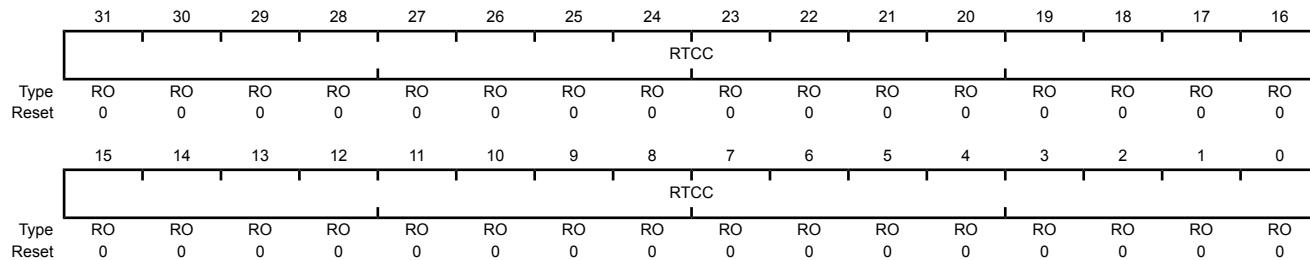
Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See “Register Access Timing” on page 304.

Hibernation RTC Counter (HIBRTCC)

Base 0x400F.C000

Offset 0x000

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	RTCC	RO	0x0000.0000	RTC Counter A read returns the 32-bit counter value, which represents the seconds elapsed since the RTC was enabled. This register is read-only. To change the value, use the HIBRTCLD register.

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

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

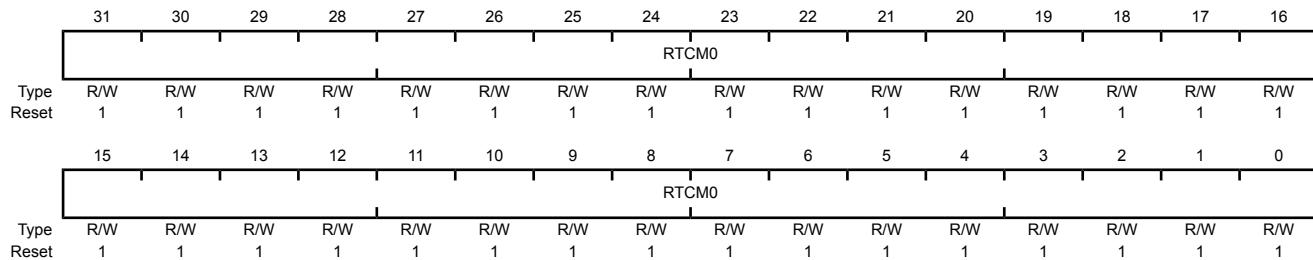
Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See “Register Access Timing” on page 304.

Hibernation RTC Match 0 (HIBRTCM0)

Base 0x400F.C000

Offset 0x004

Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	RTCM0	R/W	0xFFFF.FFFF	RTC Match 0 A write loads the value into the RTC match register. A read returns the current match value.

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

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

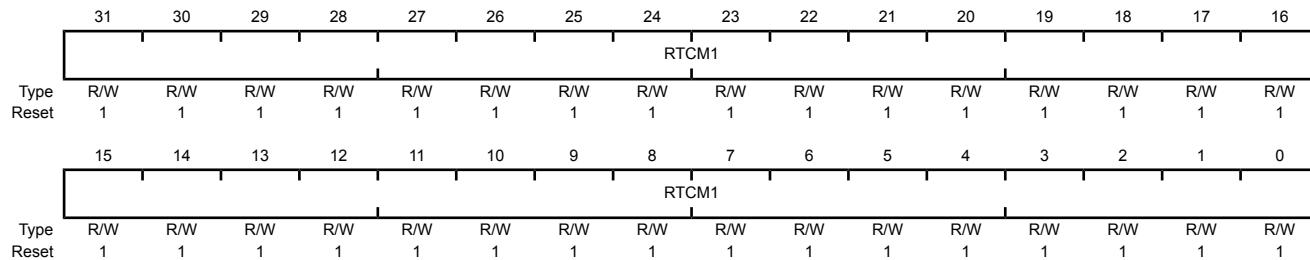
Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. If the WRC bit is clear, any attempted write access is ignored. See “Register Access Timing” on page 304.

Hibernation RTC Match 1 (HIBRTCM1)

Base 0x400F.C000

Offset 0x008

Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	RTCM1	R/W	0xFFFF.FFFF	RTC Match 1 A write loads the value into the RTC match register. A read returns the current match value.

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

This register is used to load a 32-bit value loaded into the RTC counter. The load occurs immediately upon this register being written.

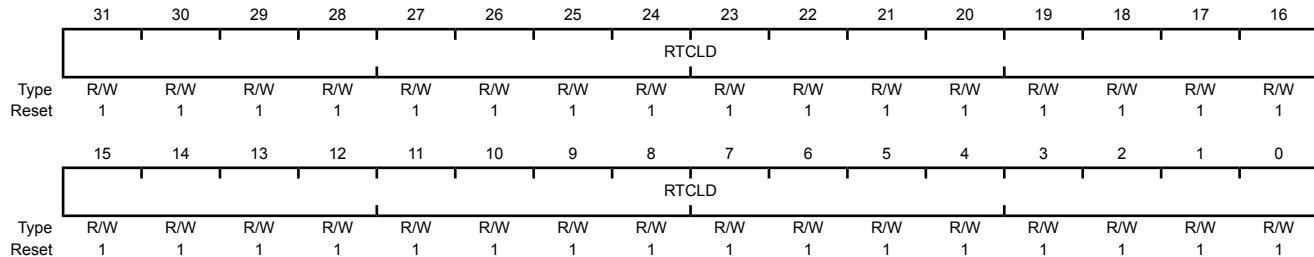
Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the **WRC** bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. If the **WRC** bit is clear, any attempted write access is ignored. See “Register Access Timing” on page 304.

Hibernation RTC Load (HIBRTCLD)

Base 0x400F.C000

Offset 0x00C

Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	RTCLD	R/W	0xFFFF.FFFF	RTC Load A write loads the current value into the RTC counter (RTCC). A read returns the 32-bit load value.

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

This register is the control register for the Hibernation module. This register must be written last before a hibernate event is issued. Writes to other registers after the HIBREQ bit is set are not guaranteed to complete before hibernation is entered.

Hibernation Control (HIBCTL)

Base 0x400F.C000
Offset 0x010
Type R/W, reset 0x8000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	WRC	reserved														
Reset	1	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	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	WRC	RO	1	Write Complete/Capable
		Value	Description	
	0	The interface is processing a prior write and is busy. Any write operation that is attempted while WRC is 0 results in undetermined behavior.		
	1	The interface is ready to accept a write.		
30:9	reserved	RO	0x000	Software must poll this bit between write requests and defer writes until WRC=1 to ensure proper operation. The bit name WRC means "Write Complete," which is the normal use of the bit (between write accesses). However, because the bit is set out-of-reset, the name can also mean "Write Capable" which simply indicates that the interface may be written to by software. This difference may be exploited by software at reset time to detect which method of programming is appropriate: 0 = software delay loops required; 1 = WRC paced available.
8	VDD3ON	R/W	0	VDD Powered
		Value	Description	
	1	The internal switches control the power to the on-chip modules (VDD3ON mode).		
	0	The internal switches are not used. The \overline{HIB} signal should be used to control an external switch or regulator.		
		Note		Note that regardless of the status of the VDD3ON bit, the \overline{HIB} signal is asserted during Hibernate mode. Thus, when VDD3ON is set, the \overline{HIB} signal should not be connected to the 3.3V regulator, and the 3.3V power source should remain connected.

Bit/Field	Name	Type	Reset	Description	
7	VABORT	R/W	0	Power Cut Abort Enable	
				Value	Description
				1	When this bit is set, the battery voltage level is checked before entering hibernation. If V_{BAT} is less than V_{LOWBAT} , the microcontroller does not go into hibernation.
				0	The microcontroller goes into hibernation regardless of the voltage level of the battery.
6	CLK32EN	R/W	0	Clocking Enable This bit must be enabled to use the Hibernation module.	
				Value	Description
				1	The Hibernation module clock source is enabled.
				0	The Hibernation module clock source is disabled.
5	LOWBATEN	R/W	0	Low Battery Monitoring Enable	
				Value	Description
				1	Low battery voltage detection is enabled. When this bit is set, the battery voltage level is checked before entering hibernation. If V_{BAT} is less than V_{LOWBAT} , the LOWBAT bit in the HIBRIS register is set.
				0	Low battery monitoring is disabled.
4	PINWEN	R/W	0	External WAKE Pin Enable	
				Value	Description
				1	An assertion of the WAKE pin takes the microcontroller out of hibernation.
				0	The status of the WAKE pin has no effect on hibernation.
3	RTCWEN	R/W	0	RTC Wake-up Enable	
				Value	Description
				1	An RTC match event (the value the HIBRTCC register matches the value of the HIBRTCM0 or HIBRTCM1 register) takes the microcontroller out of hibernation.
				0	An RTC match event has no effect on hibernation.
2	CLKSEL	R/W	0	Hibernation Module Clock Select	
				Value	Description
				1	Use raw output. Use this value for a 32.768-kHz oscillator.
				0	Use Divide-by-128 output. Use this value for a 4.194304-MHz crystal.

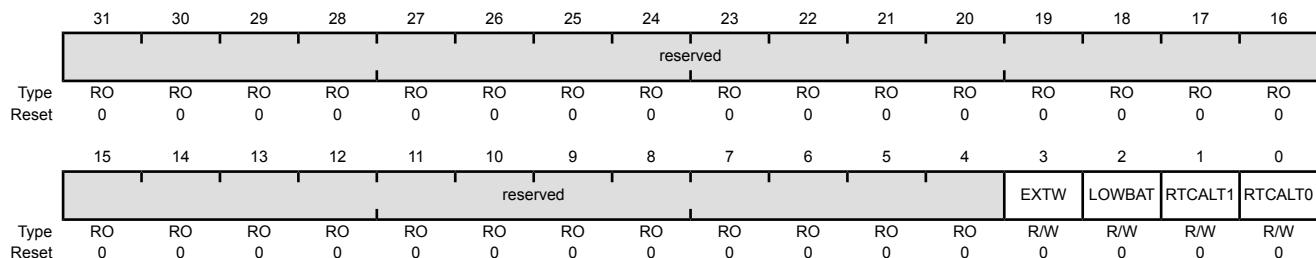
Bit/Field	Name	Type	Reset	Description
1	HIBREQ	R/W	0	Hibernation Request
				Value Description
				1 Set this bit to initiate hibernation.
				0 No hibernation request.
				After a wake-up event, this bit is automatically cleared by hardware.
0	RTCEN	R/W	0	RTC Timer Enable
				Value Description
				1 The Hibernation module RTC is enabled. The RTC remains active during hibernation.
				0 The Hibernation module RTC is disabled. When this bit is clear and <code>PINWEN</code> is set, enabling an external wake event, the RTC stops during hibernation to save power.

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

This register is the interrupt mask register for the Hibernation module interrupt sources. Each bit in this register masks the corresponding bit in the **Hibernation Raw Interrupt Status (HIBRIS)** register. If a bit is unmasked, the interrupt is sent to the interrupt controller. If the bit is masked, the interrupt is not sent to the interrupt controller.

Hibernation Interrupt Mask (HIBIM)

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



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	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	EXTW	R/W	0	External Wake-Up Interrupt Mask
		Value	Description	
		1	An interrupt is sent to the interrupt controller when the EXTW bit in the HIBRIS register is set.	
		0	The EXTW interrupt is suppressed and not sent to the interrupt controller.	
2	LOWBAT	R/W	0	Low Battery Voltage Interrupt Mask
		Value	Description	
		1	An interrupt is sent to the interrupt controller when the LOWBAT bit in the HIBRIS register is set.	
		0	The LOWBAT interrupt is suppressed and not sent to the interrupt controller.	
1	RTCACT1	R/W	0	RTC Alert 1 Interrupt Mask
		Value	Description	
		1	An interrupt is sent to the interrupt controller when the RTCACT1 bit in the HIBRIS register is set.	
		0	The RTCACT1 interrupt is suppressed and not sent to the interrupt controller.	

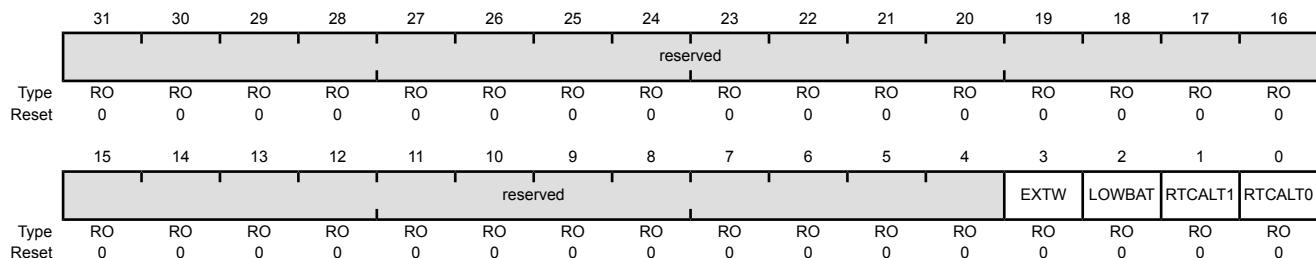
Bit/Field	Name	Type	Reset	Description
0	RTCALT0	R/W	0	RTC Alert 0 Interrupt Mask
Value Description				
			1	An interrupt is sent to the interrupt controller when the RTCALT0 bit in the HIBRIS register is set.
			0	The RTCALT0 interrupt is suppressed and not sent to the interrupt controller.

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

This register is the raw interrupt status for the Hibernation module interrupt sources. Each bit can be masked by clearing the corresponding bit in the **HIBIM** register. When a bit is masked, the interrupt is not sent to the interrupt controller. Bits in this register are cleared by writing a 1 to the corresponding bit in the **Hibernation Interrupt Clear (HIBIC)** register.

Hibernation Raw Interrupt Status (HIBRIS)

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



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	RO	0	External Wake-Up Raw Interrupt Status
	Value	Description		
	1	The \overline{WAKE} pin has been asserted.		
	0	The \overline{WAKE} pin has not been asserted.		
	This bit is cleared by writing a 1 to the EXTW bit in the HIBIC register.			
2	LOWBAT	RO	0	Low Battery Voltage Raw Interrupt Status
	Value	Description		
	1	The battery voltage dropped below V_{LOWBAT} .		
	0	The battery voltage has not dropped below V_{LOWBAT} .		
	This bit is cleared by writing a 1 to the LOWBAT bit in the HIBIC register.			
1	RTCALT1	RO	0	RTC Alert 1 Raw Interrupt Status
	Value	Description		
	1	The value of the HIBRTCC register matches the value in the HIBRTCM1 register.		
	0	No match		
	This bit is cleared by writing a 1 to the RTCALT1 bit in the HIBIC register.			

Bit/Field	Name	Type	Reset	Description						
0	RTCALT0	RO	0	RTC Alert 0 Raw Interrupt Status						
<table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>1</td><td>The value of the HIBRTCC register matches the value in the HIBRTCM0 register.</td></tr><tr><td>0</td><td>No match</td></tr></tbody></table>					Value	Description	1	The value of the HIBRTCC register matches the value in the HIBRTCM0 register.	0	No match
Value	Description									
1	The value of the HIBRTCC register matches the value in the HIBRTCM0 register.									
0	No match									
This bit is cleared by writing a 1 to the RTCALT0 bit in the HIBIC register.										

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

This register is the masked interrupt status for the Hibernation module interrupt sources. Bits in this register are the AND of the corresponding bits in the **HIBRIS** and **HIBIM** registers. When both corresponding bits are set, the bit in this register is set, and the interrupt is sent to the interrupt controller.

Hibernation Masked Interrupt Status (HIBMIS)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	RO	0	External Wake-Up Masked Interrupt Status
		Value	Description	
		1	An unmasked interrupt was signaled due to a <code>WAKE</code> pin assertion.	
		0	An external wake-up interrupt has not occurred or is masked.	
		This bit is cleared by writing a 1 to the <code>EXTW</code> bit in the HIBIC register.		
2	LOWBAT	RO	0	Low Battery Voltage Masked Interrupt Status
		Value	Description	
		1	An unmasked interrupt was signaled due to a low battery voltage condition.	
		0	A low battery voltage interrupt has not occurred or is masked.	
		This bit is cleared by writing a 1 to the <code>LOWBAT</code> bit in the HIBIC register.		
1	RTCAWT1	RO	0	RTC Alert 1 Masked Interrupt Status
		Value	Description	
		1	An unmasked interrupt was signaled due to an RTC match.	
		0	An RTC match interrupt has not occurred or is masked.	
		This bit is cleared by writing a 1 to the <code>RTCAWT1</code> bit in the HIBIC register.		

Bit/Field	Name	Type	Reset	Description
0	RTCALT0	RO	0	RTC Alert 0 Masked Interrupt Status
Value Description				
1 An unmasked interrupt was signaled due to an RTC match. 0 An RTC match interrupt has not occurred or is masked.				

This bit is cleared by writing a 1 to the RTCALT0 bit in the **HIBIC** register.

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

This register is the interrupt write-one-to-clear register for the Hibernation module interrupt sources. Writing a 1 to a bit clears the corresponding interrupt in the **HIBRIS** register.

Hibernation Interrupt Clear (HIBIC)

Base 0x400F.C000
Offset 0x020
Type R/W1C, reset 0x0000.0000

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	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	EXTW	R/W1C	0	External Wake-Up Masked Interrupt Clear Writing a 1 to this bit clears the EXTW bit in the HIBRIS and HIBMIS registers. Reads return an indeterminate value.
2	LOWBAT	R/W1C	0	Low Battery Voltage Masked Interrupt Clear Writing a 1 to this bit clears the LOWBAT bit in the HIBRIS and HIBMIS registers. Reads return an indeterminate value.
1	RTCALT1	R/W1C	0	RTC Alert1 Masked Interrupt Clear Writing a 1 to this bit clears the RTCALT1 bit in the HIBRIS and HIBMIS registers. Reads return an indeterminate value.
0	RTCALT0	R/W1C	0	RTC Alert0 Masked Interrupt Clear Writing a 1 to this bit clears the RTCALT0 bit in the HIBRIS and HIBMIS registers. Reads return an indeterminate value.

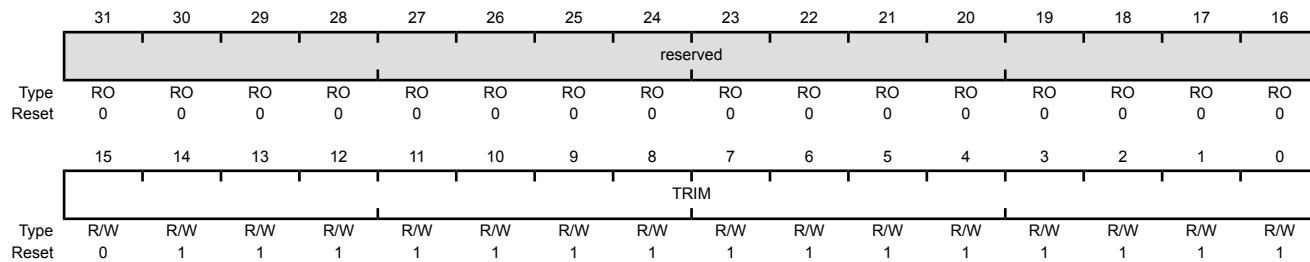
Register 10: Hibernation RTC Trim (HIBRTCT), offset 0x024

This register contains the value that is used to trim the RTC clock predivider. It represents the computed underflow value that is used during the trim cycle. It is represented as $0x7FFF \pm N$ clock cycles, where N is the number of clock cycles to add or subtract every 63 seconds.

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the **WRC** bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. If the **WRC** bit is clear, any attempted write access is ignored. See “Register Access Timing” on page 304.

Hibernation RTC Trim (HIBRTCT)

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



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

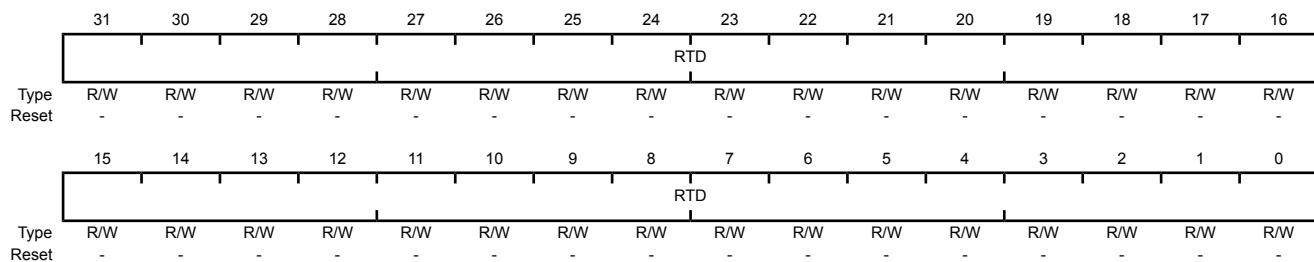
Register 11: Hibernation Data (HIBDATA), offset 0x030-0x12C

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

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the **WRC** bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. If the **WRC** bit is clear, any attempted write access is ignored. See “Register Access Timing” on page 304.

Hibernation Data (HIBDATA)

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



Bit/Field	Name	Type	Reset	Description
31:0	RTD	R/W	-	Hibernation Module NV Data

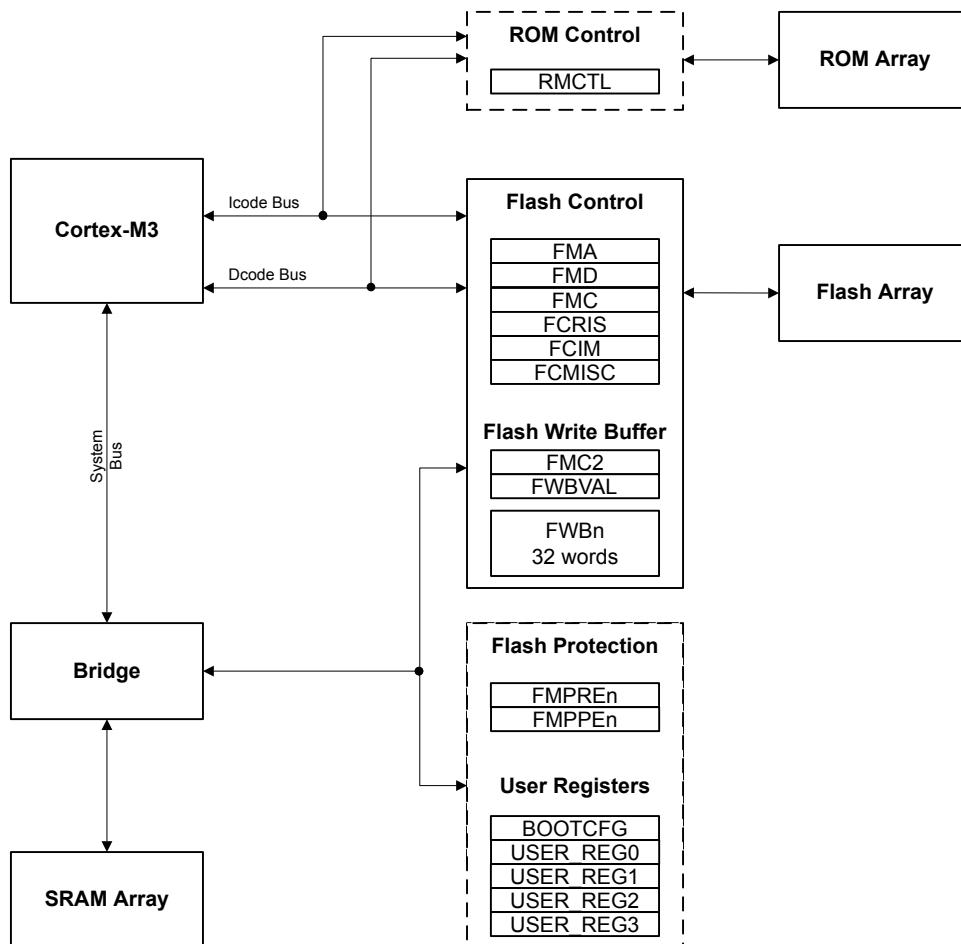
7 Internal Memory

The LM3S9L97 microcontroller comes with 48 KB of bit-banded SRAM, internal ROM, and 128 KB of Flash memory. The Flash memory controller provides a user-friendly interface, making Flash memory programming a simple task. Flash memory protection can be applied to the Flash memory on a 2-KB block basis.

7.1 Block Diagram

Figure 7-1 on page 328 illustrates the internal memory blocks and control logic. The dashed boxes in the figure indicate registers residing in the System Control module.

Figure 7-1. Internal Memory Block Diagram



7.2 Functional Description

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

Note: The µDMA controller can transfer data to and from the on-chip SRAM. However, because the Flash memory and ROM are located on a separate internal bus, it is not possible to transfer data from the Flash memory or ROM with the µDMA controller.

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 provides bit-banding technology in the 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:

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

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

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

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

For details about bit-banding, see “Bit-Banding” on page 97.

7.2.2 ROM

The internal ROM of the Stellaris device is located at address 0x0100.0000 of the device memory map. Detailed information on the ROM contents can be found in the *Stellaris® ROM User’s Guide*.

The ROM contains the following components:

- Stellaris Boot Loader and vector table
- Stellaris Peripheral Driver Library (DriverLib) release for product-specific peripherals and interfaces
- Advanced Encryption Standard (AES) cryptography tables
- Cyclic Redundancy Check (CRC) error detection functionality

The boot loader is used as an initial program loader (when the Flash memory is empty) as well as an application-initiated firmware upgrade mechanism (by calling back to the boot loader). The Peripheral Driver Library APIs in ROM can be called by applications, reducing Flash memory requirements and freeing the Flash memory to be used for other purposes (such as additional features in the application). Advance Encryption Standard (AES) is a publicly defined encryption standard used by the U.S. Government and Cyclic Redundancy Check (CRC) is a technique to validate a span of data has the same contents as when previously checked.

7.2.2.1 Boot Loader Overview

The Stellaris Boot Loader is used to download code to the Flash memory of a device without the use of a debug interface. When the core is reset, the user has the opportunity to direct the core to execute the ROM Boot Loader or the application in Flash memory by using any GPIO signal in Ports A-H as configured in the **Boot Configuration (BOOTCFG)** register.

At reset, the ROM is mapped over the Flash memory so that the ROM boot sequence is always executed. The boot sequence executed from ROM is as follows:

1. The **BA** bit (below) is cleared such that ROM is mapped to 0x01xx.xxxx and Flash memory is mapped to address 0x0.
2. The **BOOTCFG** register is read. If the **EN** bit is set, the status of the specified GPIO pin is compared with the specified polarity. If the status matches the specified polarity, the ROM Boot Loader is executed.
3. If the status doesn't match the specified polarity, the address at address 0x0000.0004 is checked to see if the Flash memory is programmed. If the data at this address is 0xFFFF.FFFF, the ROM Boot Loader is executed.
4. If there is valid data at address 0x0000.0004, the stack pointer (**SP**) is loaded from Flash memory at address 0x0000.0000 and the program counter (**PC**) is loaded from address 0x0000.0004. The user application begins executing.

The boot loader uses a simple packet interface to provide synchronous communication with the device. The speed of the boot loader is determined by the internal oscillator (PIOSC) frequency as it does not enable the PLL. The following serial interfaces can be used:

- UART0
- SSI0
- I²C0
- Ethernet

For simplicity, both the data format and communication protocol are identical for all serial interfaces.

Note: The Flash-memory-resident version of the Boot Loader also supports CAN and USB.

See the *Stellaris® Boot Loader User's Guide* for information on the boot loader software.

7.2.2.2 Stellaris Peripheral Driver Library

The Stellaris Peripheral Driver Library contains a file called `driverlib/rom.h` that assists with calling the peripheral driver library functions in the ROM. The detailed description of each function is available in the *Stellaris® ROM User's Guide*. See the "Using the ROM" chapter of the *Stellaris® Peripheral Driver Library User's Guide* for more details on calling the ROM functions and using `driverlib/rom.h`.

A table at the beginning of the ROM points to the entry points for the APIs that are provided in the ROM. Accessing the API through these tables provides scalability; while the API locations may change in future versions of the ROM, the API tables will not. The tables are split into two levels; the main table contains one pointer per peripheral which points to a secondary table that contains one pointer per API that is associated with that peripheral. The main table is located at 0x0100.0010, right after the Cortex-M3 vector table in the ROM.

DriverLib functions are described in detail in the *Stellaris® Peripheral Driver Library User's Guide*.

Additional APIs are available for graphics and USB functions, but are not preloaded into ROM. The Stellaris Graphics Library provides a set of graphics primitives and a widget set for creating graphical user interfaces on Stellaris microcontroller-based boards that have a graphical display (for more information, see the *Stellaris® Graphics Library User's Guide*). The Stellaris USB Library is a set

of data types and functions for creating USB Device, Host or On-The-Go (OTG) applications on Stellaris microcontroller-based boards (for more information, see the *Stellaris® USB Library User's Guide*).

7.2.2.3 Advanced Encryption Standard (AES) Cryptography Tables

AES 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. Four data tables used by the XySSL AES implementation are provided in the ROM. The first is the forward S-box substitution table, the second is the reverse S-box substitution table, the third is the forward polynomial table, and the final is the reverse polynomial table. See the *Stellaris® ROM User's Guide* for more information on AES.

7.2.2.4 Cyclic Redundancy Check (CRC) Error Detection

The CRC 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. See the *Stellaris® ROM User's Guide* for more information on CRC.

7.2.3 Flash Memory

At system clock speeds of 50 MHz and below, the Flash memory is read in a single cycle. The Flash memory 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.

Caution – The Stellaris Flash memory array has ECC which uses a test port into the Flash memory to continually scan the array for ECC errors and to correct any that are detected. This operation is transparent to the microcontroller. The BIST must scan the entire memory array occasionally to ensure integrity, taking about five minutes to do so. In systems where the microcontroller is frequently powered for less than five minutes, power should be removed from the microcontroller in a controlled manner to ensure proper operation. This controlled manner can either be through entering Hibernation mode or software can request permission to power down the part using the USDREQ bit in the Flash Control (FCTL) register and wait to receive an acknowledge from the USDACK bit prior to removing power. If the microcontroller is powered down using this controlled method, the BIST engine keeps track of where it was in the memory array and it always scans the complete array after any aggregate of five minutes powered-on, regardless of the number of intervening power cycles. If the microcontroller is powered down before five minutes of being powered up, BIST starts again from wherever it left off before the last controlled power-down or from 0 if there never was a controlled power down. An occasional short power down is not a concern, but the microcontroller should not always be powered down frequently in an uncontrolled manner. The microcontroller can be power-cycled as frequently as necessary if it is powered-down in a controlled manner.

7.2.3.1 Prefetch Buffer

The Flash memory controller has a prefetch buffer that is automatically used when the CPU frequency is greater than 50 MHz. In this mode, the Flash memory operates at half of the system clock. The prefetch buffer fetches two 32-bit words per clock allowing instructions to be fetched with no wait states while code is executing linearly. The fetch buffer includes a branch speculation mechanism that recognizes a branch and avoids extra wait states by not reading the next word pair. Also, short loop branches often stay in the buffer. As a result, some branches can be executed with no wait states. Other branches incur a single wait state.

7.2.3.2 Flash Memory Protection

The user is provided two forms of Flash memory protection per 2-KB Flash memory 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 read as data.

The policies may be combined as shown in Table 7-1 on page 332.

Table 7-1. Flash Memory 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.

A Flash memory access that attempts to read a read-protected block (**FMPREn** bit is set) is prohibited and generates a bus fault. A Flash memory access that attempts to program or erase a program-protected block (**FMPPEn** bit is set) is prohibited and can optionally generate an interrupt (by setting the **AMASK** bit in the **Flash Controller Interrupt Mask (FCIM)** register) to alert software developers of poorly behaving software during the development and debug phases.

The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. These settings create a policy of open access and programmability. The register bits may be changed by clearing the specific register bit. The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The changes are committed using the **Flash Memory Control (FMC)** register. Details on programming these bits are discussed in “Nonvolatile Register Programming” on page 335.

7.2.3.3 Interrupts

The Flash memory controller can generate interrupts when the following conditions are observed:

- Programming Interrupt - signals when a program or erase action is complete.
- Access Interrupt - signals when a program or erase action has been attempted on a 2-kB block of memory that is protected by its corresponding **FMPPEn** bit.

The interrupt events that can trigger a controller-level interrupt are defined in the **Flash Controller Masked Interrupt Status (FCMIS)** register (see page 344) by setting the corresponding MASK bits. If interrupts are not used, the raw interrupt status is always visible via the **Flash Controller Raw Interrupt Status (FCRIS)** register (see page 343).

Interrupts are always cleared (for both the **FCMIS** and **FCRIS** registers) by writing a 1 to the corresponding bit in the **Flash Controller Masked Interrupt Status and Clear (FCMISC)** register (see page 345).

7.3 Flash Memory Initialization and Configuration

7.3.1 Flash Memory Programming

The Stellaris devices provide a user-friendly interface for Flash memory 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 188.

During a Flash memory operation (write, page erase, or mass erase) access to the Flash memory is inhibited. As a result, instruction and literal fetches are held off until the Flash memory operation is complete. If instruction execution is required during a Flash memory operation, the code that is executing must be placed in SRAM and executed from there while the flash operation is in progress.

Caution – The Flash memory is divided into sectors of electrically separated address ranges of 4 KB each, aligned on 4 KB boundaries. Erase/program operations on a 1-KB page have an electrical effect on the other three 1-KB pages within the sector. A specific 1-KB page must be erased after 6 total erase/program cycles occur to the other pages within it's 4-KB sector. The following sequence of operations on a 4-KB sector of Flash memory (Page 0..3) provides an example:

- Page 3 is erase and programmed with values.
- Page 0, Page 1, and Page 2 are erased and then programmed with values. At this point Page 3 has been affected by 3 erase/program cycles.
- Page 0, Page 1, and Page 2 are again erased and then programmed with values. At this point Page 3 has been affected by 6 erase/program cycles.
- If the contents of Page 3 must continue to be valid, Page 3 must be erased and reprogrammed before any other page in this sector has another erase or program operation.

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 memory 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`. The following two sequences are allowed:

- `ERASE` -> `PROGRAM` value -> `PROGRAM` 0x0000.0000
- `ERASE` -> `PROGRAM` value -> `ERASE`

The following sequence is NOT allowed:

- `ERASE` -> `PROGRAM` value -> `PROGRAM` value

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 memory 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 or, alternatively, enable the programming interrupt using the `PMASK` bit in the **FCIM** register.

7.3.1.3 To perform a mass erase of the Flash memory

1. Write the Flash memory 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 or, alternatively, enable the programming interrupt using the `PMASK` bit in the **FCIM** register.

7.3.2 32-Word Flash Memory 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 memory write operation. The buffered Flash memory 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 memory write operation are written. The **Flash Write Buffer Valid (FWBVAL)** register shows which registers have been written since the last buffered Flash memory 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 memory 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 memory 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 or wait for the **PMIS** interrupt to be signaled.

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 memory array and are not affected by an ERASE or MASS ERASE operation. The bits in these registers can be changed from 1 to 0 with a write operation. The register contents are unaffected by any reset condition except power-on reset, which returns the register contents to 0xFFFF.FFFF. By committing the register values using the **COMT** bit in the **FMC** register, the register contents become nonvolatile and are therefore retained following power cycling. Once the register contents are committed, the only way to restore the factory default values is to perform the sequence described in “Recovering a “Locked” Microcontroller” on page 188.

With the exception of the **Boot Configuration (BOOTCFG)** register, the settings in these registers can be tested before committing them to Flash memory. For the **BOOTCFG** 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 **BOOTCFG** operation to be tried before committing it to nonvolatile memory.

Important: The Flash memory resident registers can only have bits changed from 1 to 0 by user programming and can only be committed once. After being committed, these registers can only be restored to their factory default values only by performing the sequence described in “Recovering a “Locked” Microcontroller” on page 188. The mass erase of the main Flash memory array caused by the sequence is performed prior to restoring these registers.

In addition, the **USER_REG0**, **USER_REG1**, **USER_REG2**, **USER_REG3**, and **BOOTCFG** registers each use bit 31 (**NW**) to indicate that they have not been committed and bits in the register may be changed from 1 to 0. Table 7-2 on page 335 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 Memory 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
BOOTCFG	0x7510.0000	FMD

7.4 Register Map

Table 7-3 on page 336 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 memory control base address of 0x400F.D000. The ROM and Flash memory 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
Flash Memory Registers (Flash Control Offset)					
0x000	FMA	R/W	0x0000.0000	Flash Memory Address	338
0x004	FMD	R/W	0x0000.0000	Flash Memory Data	339
0x008	FMC	R/W	0x0000.0000	Flash Memory Control	340
0x00C	FCRIS	RO	0x0000.0000	Flash Controller Raw Interrupt Status	343
0x010	FCIM	R/W	0x0000.0000	Flash Controller Interrupt Mask	344
0x014	FCMISC	R/W1C	0x0000.0000	Flash Controller Masked Interrupt Status and Clear	345
0x020	FMC2	R/W	0x0000.0000	Flash Memory Control 2	346
0x030	FWBVAL	R/W	0x0000.0000	Flash Write Buffer Valid	347
0x0F8	FCTL	R/W	0x0000.0000	Flash Control	348
0x100 - 0x17C	FWBn	R/W	0x0000.0000	Flash Write Buffer n	349
Memory Registers (System Control Offset)					
0x0F0	RMCTL	R/W1C	-	ROM Control	350
0x130	FMPRE0	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 0	351
0x200	FMPRE0	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 0	351
0x134	FMPPE0	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 0	352
0x400	FMPPE0	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 0	352
0x1D0	BOOTCFG	R/W	0xFFFF.FFFE	Boot Configuration	353
0x1E0	USER_REG0	R/W	0xFFFF.FFFF	User Register 0	355
0x1E4	USER_REG1	R/W	0xFFFF.FFFF	User Register 1	356
0x1E8	USER_REG2	R/W	0xFFFF.FFFF	User Register 2	357
0x1EC	USER_REG3	R/W	0xFFFF.FFFF	User Register 3	358
0x204	FMPRE1	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 1	359
0x208	FMPRE2	R/W	0x0000.0000	Flash Memory Protection Read Enable 2	360
0x20C	FMPRE3	R/W	0x0000.0000	Flash Memory Protection Read Enable 3	361
0x404	FMPPE1	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 1	362

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

Offset	Name	Type	Reset	Description	See page
0x408	FMPPE2	R/W	0x0000.0000	Flash Memory Protection Program Enable 2	363
0x40C	FMPPE3	R/W	0x0000.0000	Flash Memory Protection Program Enable 3	364

7.5 Flash Memory Register Descriptions (Flash Control Offset)

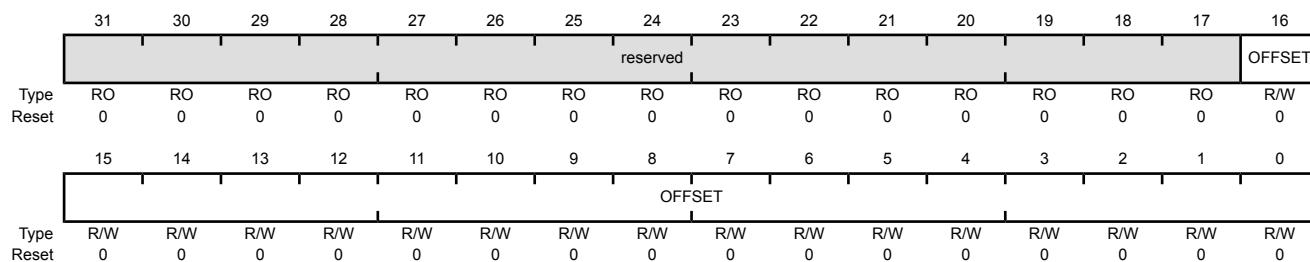
This section lists and describes the Flash Memory registers, in numerical order by address offset. Registers in this section are relative to the Flash control base address of 0x400F.D000.

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 CPU byte 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



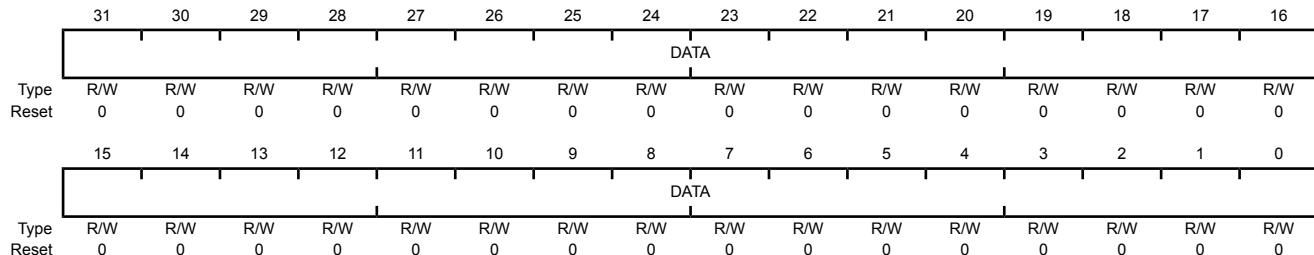
Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16:0	OFFSET	R/W	0x0	Address Offset Address offset in Flash memory where operation is performed, except for nonvolatile registers (see "Nonvolatile Register Programming" on page 335 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



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 memory controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 338). If the access is a write access, the data contained in the **Flash Memory Data (FMD)** register (see page 339) 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.

Caution – If any of bits [15:4] are written to 1, the device may become inoperable. These bits should always be written to 0. In all registers, the value of a reserved bit should be preserved across a read-modify-write operation.

Flash Memory Control (FMC)

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

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

Bit/Field	Name	Type	Reset	Description
31:16	WRKEY	WO	0x0000	Flash Memory Write Key This field contains a write key, which is used to minimize the incidence of accidental Flash memory writes. The value 0xA442 must be written into this field for a Flash memory write to occur. Writes to the FMC register without this WRKEY 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.

Bit/Field	Name	Type	Reset	Description						
3	COMT	R/W	0	<p>Commit Register Value</p> <p>This bit is used to commit writes to Flash-memory-resident registers and to monitor the progress of that process.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Set this bit to commit (write) the register value to a Flash-memory-resident register. When read, a 1 indicates that the previous commit access is not complete.</td></tr> <tr> <td>0</td><td>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.</td></tr> </tbody> </table> <p>A commit can take up to 50 µs. See “Nonvolatile Register Programming” on page 335 for more information on programming Flash-memory-resident registers.</p>	Value	Description	1	Set this bit to commit (write) the register value to a Flash-memory-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.
Value	Description									
1	Set this bit to commit (write) the register value to a Flash-memory-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.									
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> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Set this bit to erase the Flash main memory. When read, a 1 indicates that the previous mass erase access is not complete.</td></tr> <tr> <td>0</td><td>A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous mass erase access is complete.</td></tr> </tbody> </table> <p>A mass erase can take up to 16 ms.</p>	Value	Description	1	Set this bit to erase the Flash main memory. When read, a 1 indicates that the previous mass erase 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 mass erase access is complete.
Value	Description									
1	Set this bit to erase the Flash main memory. When read, a 1 indicates that the previous mass erase 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 mass erase access is complete.									
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> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Set this bit to erase the Flash memory page specified by the contents of the FMA register. When read, a 1 indicates that the previous page erase access is not complete.</td></tr> <tr> <td>0</td><td>A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous page erase access is complete.</td></tr> </tbody> </table> <p>A page erase can take up to 25 ms.</p>	Value	Description	1	Set this bit to erase the Flash memory page specified by the contents of the FMA register. When read, a 1 indicates that the previous page erase 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 page erase access is complete.
Value	Description									
1	Set this bit to erase the Flash memory page specified by the contents of the FMA register. When read, a 1 indicates that the previous page erase 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 page erase access is complete.									

Bit/Field	Name	Type	Reset	Description
0	WRITE	R/W	0	<p>Write a Word into Flash Memory This bit is used to write a word into Flash memory and to monitor the progress of that process.</p>
Value Description				
1 Set this bit to write the data stored in the FMD register into the Flash memory location specified by the contents of the FMA register. When read, a 1 indicates that the write update 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 write update access is complete.				

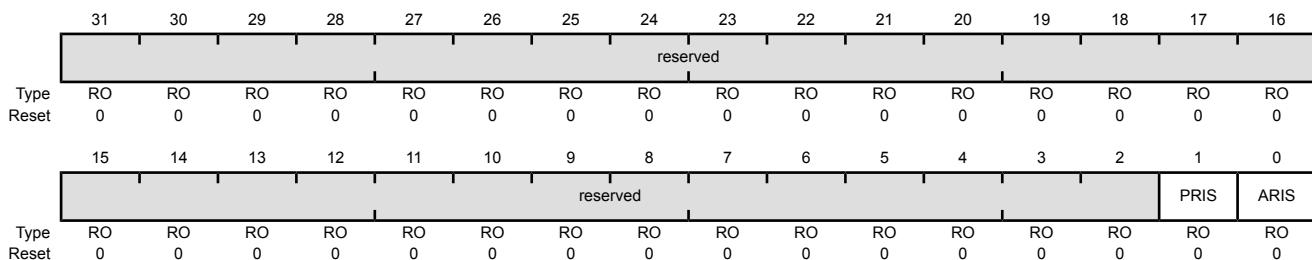
Writing a single word can take up to 50 μ s.

Register 4: Flash Controller Raw Interrupt Status (FCRIS), offset 0x00C

This register indicates that the Flash memory 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



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 Programming Raw Interrupt Status
This bit provides status on programming cycles which are write or erase actions generated through the **FMC** or **FMC2** register bits (see page 340 and page 346).

Value Description

- 1 The programming or erase cycle has completed.
- 0 The programming or erase cycle has not completed.

This status is sent to the interrupt controller when the **PMASK** bit in the **FCIM** register is set.

This bit is cleared by writing a 1 to the **PMISC** bit in the **FCMISC** register.

0 ARIS RO 0 Access Raw Interrupt Status

Value Description

- 1 A program or erase action was attempted on a block of Flash memory that contradicts the protection policy for that block as set in the **FMPPEn** registers.
- 0 No access has tried to improperly program or erase the Flash memory.

This status is sent to the interrupt controller when the **AMASK** bit in the **FCIM** register is set.

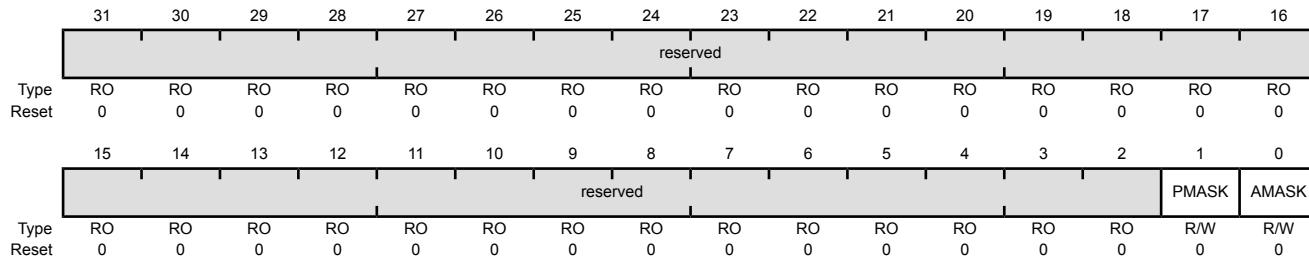
This bit is cleared by writing a 1 to the **AMISC** bit in the **FCMISC** register.

Register 5: Flash Controller Interrupt Mask (FCIM), offset 0x010

This register controls whether the Flash memory controller generates interrupts to the controller.

Flash Controller Interrupt Mask (FCIM)

Base 0x400F.D000
Offset 0x010
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:2	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
1	PMASK	R/W	0	<p>Programming Interrupt Mask</p> <p>This bit controls the reporting of the programming raw interrupt status to the interrupt controller.</p> <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An interrupt is sent to the interrupt controller when the PRIS bit is set.</td></tr> <tr> <td>0</td><td>The PRIS interrupt is suppressed and not sent to the interrupt controller.</td></tr> </tbody> </table>	Value	Description	1	An interrupt is sent to the interrupt controller when the PRIS bit is set.	0	The PRIS interrupt is suppressed and not sent to the interrupt controller.
Value	Description									
1	An interrupt is sent to the interrupt controller when the PRIS bit is set.									
0	The PRIS interrupt is suppressed and not sent to the interrupt controller.									
0	AMASK	R/W	0	<p>Access Interrupt Mask</p> <p>This bit controls the reporting of the access raw interrupt status to the interrupt controller.</p> <table border="1"> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An interrupt is sent to the interrupt controller when the ARIS bit is set.</td></tr> <tr> <td>0</td><td>The ARIS interrupt is suppressed and not sent to the interrupt controller.</td></tr> </tbody> </table>	Value	Description	1	An interrupt is sent to the interrupt controller when the ARIS bit is set.	0	The ARIS interrupt is suppressed and not sent to the interrupt controller.
Value	Description									
1	An interrupt is sent to the interrupt controller when the ARIS bit is set.									
0	The ARIS interrupt is suppressed and not sent to the interrupt controller.									

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

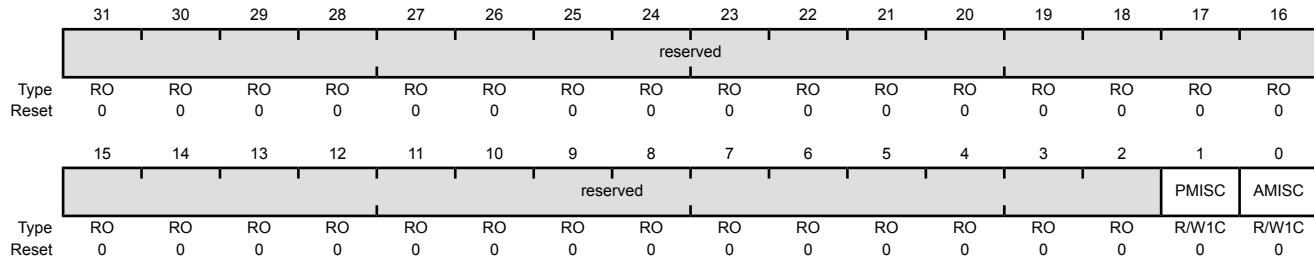
This register provides two functions. First, it reports the cause of an interrupt by indicating which interrupt source or sources are signalling the interrupt. Second, it serves as the method to clear the interrupt reporting.

Flash Controller Masked Interrupt Status and Clear (FCMISC)

Base 0x400FD000

Offset 0x014

Type R/W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PMISC	R/W1C	0	Programming Masked Interrupt Status and Clear
	Value	Description		
	1	When read, a 1 indicates that an unmasked interrupt was signaled because a programming cycle completed. Writing a 1 to this bit clears PMISC and also the PRIS bit in the FCRIS register (see page 343).		
	0	When read, a 0 indicates that a programming cycle complete interrupt has not occurred. A write of 0 has no effect on the state of this bit.		
0	AMISC	R/W1C	0	Access Masked Interrupt Status and Clear
	Value	Description		
	1	When read, a 1 indicates that an unmasked interrupt was signaled because a program or erase action was attempted on a block of Flash memory that contradicts the protection policy for that block as set in the FMPPEn registers. Writing a 1 to this bit clears AMISC and also the ARIS bit in the FCRIS register (see page 343).		
	0	When read, a 0 indicates that no improper accesses have occurred. A write of 0 has no effect on the state of this bit.		

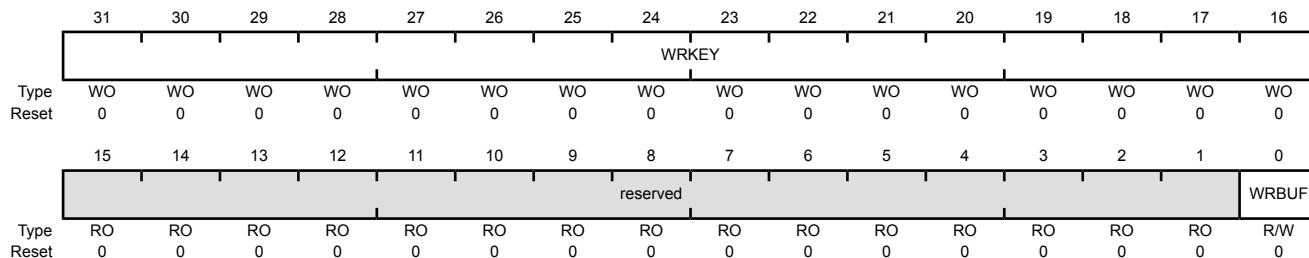
Register 7: Flash Memory Control 2 (FMC2), offset 0x020

When this register is written, the Flash memory controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 338). 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	<p>Flash Memory Write Key</p> <p>This field contains a write key, which is used to minimize the incidence of accidental Flash memory writes. The value 0xA442 must be written into this field for a write to occur. Writes to the FMC2 register without this WRKEY value are ignored. A read of this field returns the value 0.</p>
15:1	reserved	RO	0x0000	<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	WRBUF	R/W	0	<p>Buffered Flash Memory Write</p> <p>This bit is used to start a buffered write to Flash memory.</p>

Value	Description
1	<p>Set this bit to write the data stored in the FWBn registers to the location specified by the contents of the FMA register.</p> <p>When read, a 1 indicates that the previous buffered Flash memory write access is not complete.</p>
0	<p>A write of 0 has no effect on the state of this bit.</p> <p>When read, a 0 indicates that the previous buffered Flash memory write access is complete.</p>

A buffered Flash memory write can take up to 4 ms.

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

This register provides a bitwise status of which **FWBn** registers have been written by the processor since the last write of the Flash memory write buffer. The entries with a 1 are written on the next write of the Flash memory 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[n] 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 **FWBn** register change should not be written to Flash memory, software can clear the corresponding FWB[n] 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 Memory Write Buffer
				Value Description
				1 The corresponding FWBn register has been updated since the last buffer write operation and is ready to be written to Flash memory.
				0 The corresponding FWBn register has no new data to be written.
Bit 0 corresponds to FWB0 , offset 0x100, and bit 31 corresponds to FWB31 , offset 0x13C.				

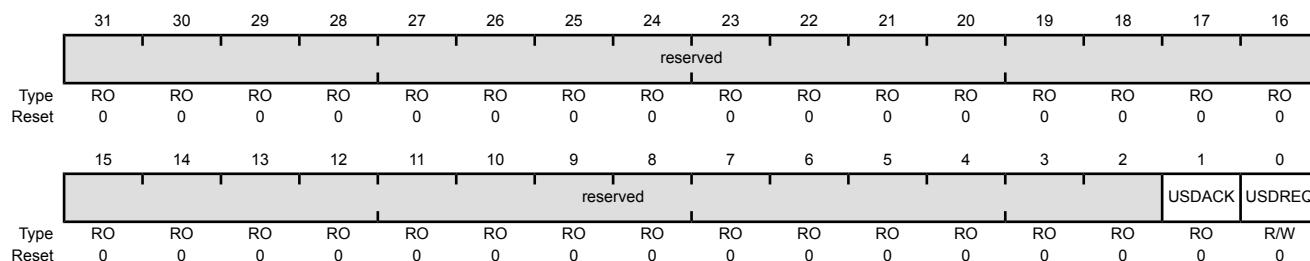
Register 9: Flash Control (FCTL), offset 0x0F8

This register is used to ensure that the microcontroller is powered down in a controlled fashion in systems where power is cycled more frequently than once every five minutes. The `USDREQ` bit should be set to indicate that power is going to be turned off. Software should poll the `USDACK` bit to determine when it is acceptable to power down.

Note that this power-down process is not required if the microcontroller enters hibernation mode prior to power being removed.

Flash Control (FCTL)

Base 0x400F.D000
Offset 0x0F8
Type R/W, reset 0x0000.0000



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	USDACK	RO	0	User Shut Down Acknowledge
		Value	Description	
		1	The microcontroller can be powered down.	
		0	The microcontroller cannot yet be powered down.	
		This bit should be set within 50 ms of setting the <code>USDREQ</code> bit.		
0	USDREQ	R/W	0	User Shut Down Request
		Value	Description	
		1	Requests permission to power down the microcontroller.	
		0	No effect.	

Register 10: Flash Write Buffer n (FWBn), offset 0x100 - 0x17C

These 32 registers hold the contents of the data to be written into the Flash memory on a buffered Flash memory write operation. The offset selects one of the 32-bit registers. Only **FWBn** registers that have been updated since the preceding buffered Flash memory write operation are written into the Flash memory, 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 memory 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 - 0x17C
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 Data to be written into the Flash memory.

7.6 Memory Register Descriptions (System Control Offset)

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

Register 11: ROM Control (RMCTL), offset 0x0F0

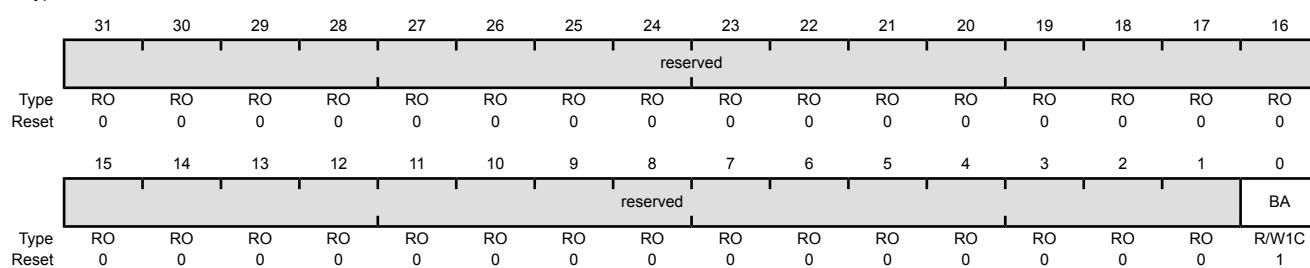
This register provides control of the ROM controller state. This register offset is relative to the System Control base address of 0x400F.E000.

At reset, the ROM is mapped over the Flash memory so that the ROM boot sequence is always executed. The boot sequence executed from ROM is as follows:

1. The **BA** bit (below) is cleared such that ROM is mapped to 0x01xx.xxxx and Flash memory is mapped to address 0x0.
2. The **BOOTCFG** register is read. If the **EN** bit is set, the status of the specified GPIO pin is compared with the specified polarity. If the status matches the specified polarity, the ROM Boot Loader is executed.
3. If the status doesn't match the specified polarity, the address at address 0x0000.0004 is checked to see if the Flash memory is programmed. If the data at this address is 0xFFFF.FFFF, the ROM Boot Loader is executed.
4. If there is valid data at address 0x0000.0004, the stack pointer (**SP**) is loaded from Flash memory at address 0x0000.0000 and the program counter (**PC**) is loaded from address 0x0000.0004. The user application begins executing.

ROM Control (RMCTL)

Base 0x400F.E000
Offset 0x0F0
Type R/W1C, reset -



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	1	Boot Alias

Value	Description
1	The microcontroller's ROM appears at address 0x0.
0	The Flash memory is at address 0x0.

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

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.

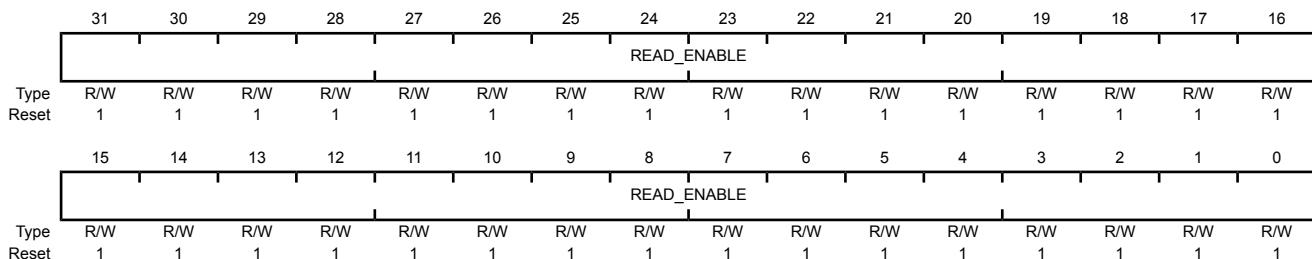
This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on 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. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. 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 Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table "Flash Protection Policy Combinations".
		Value	Description	
		0xFFFFFFFF	Bits [31:0] each enable protection on a 2-KB block of Flash memory up to the total of 64 KB.	

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

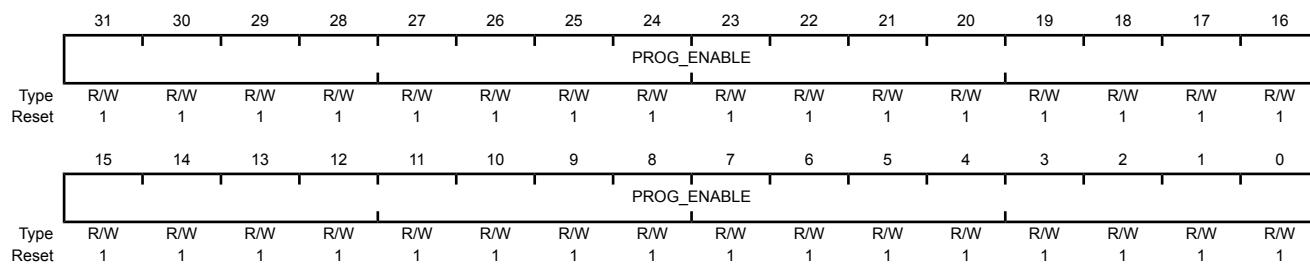
Note: This register is aliased for backwards compatibility.

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). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. 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. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. 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	Bits [31:0] each enable protection on a 2-KB block of Flash memory up to the total of 64 KB.	

Register 14: Boot Configuration (BOOTCFG), offset 0x1D0

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

This register provides configuration of a GPIO pin to enable the ROM Boot Loader as well as a write-once mechanism to disable external debugger access to the device. Upon reset, the user has the opportunity to direct the core to execute the ROM Boot Loader or the application in Flash memory by using any GPIO signal from Ports A-H as configured by the bits in this register. If the EN bit is set or the specified pin does not have the required polarity, the system control module checks address 0x000.0004 to see if the Flash memory has a valid reset vector. If the data at address 0x000.0004 is 0xFFFF.FFFF, then it is assumed that the Flash memory has not yet been programmed, and the core executes the ROM Boot Loader. 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. Clearing the DBG1 bit disables any external debugger access to the device permanently, starting with the next power-up cycle of the device. The NW bit (bit 31) indicates that the register has not yet been committed and is controlled through hardware to ensure that the register is only committed once. Prior to being committed, bits can only be changed from 1 to 0. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. The only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter.

Boot Configuration (BOOTCFG)

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								reserved							
Type	R/W	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
	PORT				PIN				POL	EN			reserved		DBG1	DBG0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	RO	RO	RO	RO	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	Not Written When set, this bit indicates that this 32-bit register has not been committed. When clear, this bit specifies that this register has been committed and may not be committed again.
30:16	reserved	RO	0x7FFF	Software should not rely on the value of 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																		
15:13	PORT	R/W	0x7	<p>Boot GPIO Port</p> <p>This field selects the port of the GPIO port pin that enables the ROM boot loader at reset.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr><td>0x0</td><td>Port A</td></tr> <tr><td>0x1</td><td>Port B</td></tr> <tr><td>0x2</td><td>Port C</td></tr> <tr><td>0x3</td><td>Port D</td></tr> <tr><td>0x4</td><td>Port E</td></tr> <tr><td>0x5</td><td>Port F</td></tr> <tr><td>0x6</td><td>Port G</td></tr> <tr><td>0x7</td><td>Port H</td></tr> </tbody> </table>	Value	Description	0x0	Port A	0x1	Port B	0x2	Port C	0x3	Port D	0x4	Port E	0x5	Port F	0x6	Port G	0x7	Port H
Value	Description																					
0x0	Port A																					
0x1	Port B																					
0x2	Port C																					
0x3	Port D																					
0x4	Port E																					
0x5	Port F																					
0x6	Port G																					
0x7	Port H																					
12:10	PIN	R/W	0x7	<p>Boot GPIO Pin</p> <p>This field selects the pin number of the GPIO port pin that enables the ROM boot loader at reset.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr><td>0x0</td><td>Pin 0</td></tr> <tr><td>0x1</td><td>Pin 1</td></tr> <tr><td>0x2</td><td>Pin 2</td></tr> <tr><td>0x3</td><td>Pin 3</td></tr> <tr><td>0x4</td><td>Pin 4</td></tr> <tr><td>0x5</td><td>Pin 5</td></tr> <tr><td>0x6</td><td>Pin 6</td></tr> <tr><td>0x7</td><td>Pin 7</td></tr> </tbody> </table>	Value	Description	0x0	Pin 0	0x1	Pin 1	0x2	Pin 2	0x3	Pin 3	0x4	Pin 4	0x5	Pin 5	0x6	Pin 6	0x7	Pin 7
Value	Description																					
0x0	Pin 0																					
0x1	Pin 1																					
0x2	Pin 2																					
0x3	Pin 3																					
0x4	Pin 4																					
0x5	Pin 5																					
0x6	Pin 6																					
0x7	Pin 7																					
9	POL	R/W	0x1	<p>Boot GPIO Polarity</p> <p>When set, this bit selects a high level for the GPIO port pin to enable the ROM boot loader at reset. When clear, this bit selects a low level for the GPIO port pin.</p>																		
8	EN	R/W	0x1	<p>Boot GPIO Enable</p> <p>Clearing this bit enables the use of a GPIO pin to enable the ROM Boot Loader at reset. When this bit is set, the contents of address 0x0000.0004 are checked to see if the Flash memory has been programmed. If the contents are not 0xFFFF.FFFF, the core executes out of Flash memory. If the Flash has not been programmed, the core executes out of ROM.</p>																		
7:2	reserved	RO	0x3F	<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	DBG1	R/W	1	<p>Debug Control 1</p> <p>The DBG1 bit must be 1 and DBG0 must be 0 for debug to be available.</p>																		
0	DBG0	R/W	0x0	<p>Debug Control 0</p> <p>The DBG1 bit must be 1 and DBG0 must be 0 for debug to be available.</p>																		

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 committed once. Bit 31 indicates that the register is available to be committed and is controlled through hardware to ensure that the register is only committed once. Prior to being committed, bits can only be changed from 1 to 0. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. 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. The only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG section.

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
	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 indicates that this 32-bit register has not been committed. When clear, this bit specifies that this register has been committed and may not be committed 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 committed once.

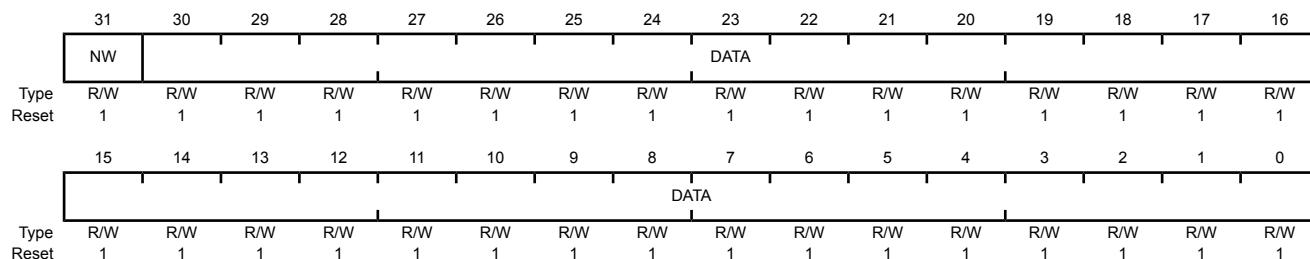
Register 16: User Register 1 (USER_REG1), offset 0x1E4

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



Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	Not Written When set, this bit indicates that this 32-bit register has not been committed. When clear, this bit specifies that this register has been committed and may not be committed 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 committed once.

Register 17: User Register 2 (USER_REG2), offset 0x1E8

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

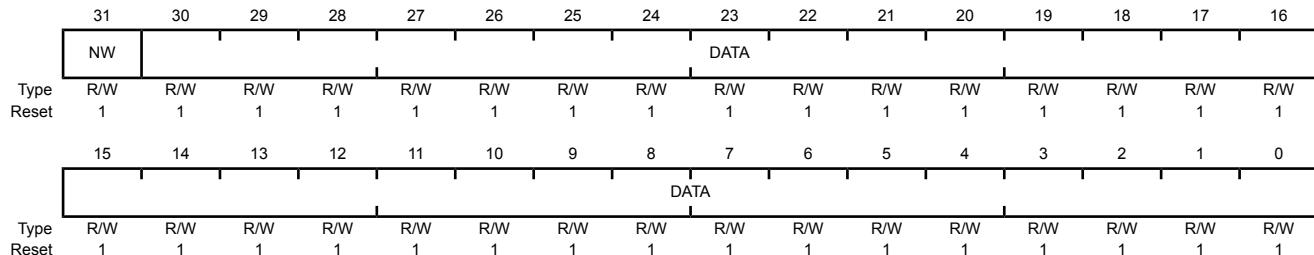
This register provides 31 bits of user-defined data that is non-volatile and can only be written once. Bit 31 indicates that the register is available to be written and is controlled through hardware to ensure that the register is only written once. The write-once characteristics of this register are 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



Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	Not Written When set, this bit indicates that this 32-bit register has not been committed. When clear, this bit specifies that this register has been committed and may not be committed 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 committed 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
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 indicates that this 32-bit register has not been committed. When clear, this bit specifies that this register has been committed and may not be committed 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 committed 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). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. 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. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 64 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. 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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
READ_ENABLE																
Type	R/W															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
READ_ENABLE																
Type	R/W															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0xFFFFFFFF	Flash Read Enable Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table "Flash Protection Policy Combinations".
				Value Description
			0xFFFFFFFF	Bits [31:0] each enable protection on a 2-KB block of Flash memory in memory range from 65 to 128 KB.

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). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. 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. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 128 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 2 (FMPRE2)

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

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

Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0x00000000	Flash Read Enable Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table "Flash Protection Policy Combinations".
				Value Description
				0x00000000 Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 129 to 192 KB.

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). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. 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. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. 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

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

Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0x00000000	Flash Read Enable Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table "Flash Protection Policy Combinations".
				Value Description
				0x00000000 Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 193 to 256 KB.

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

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

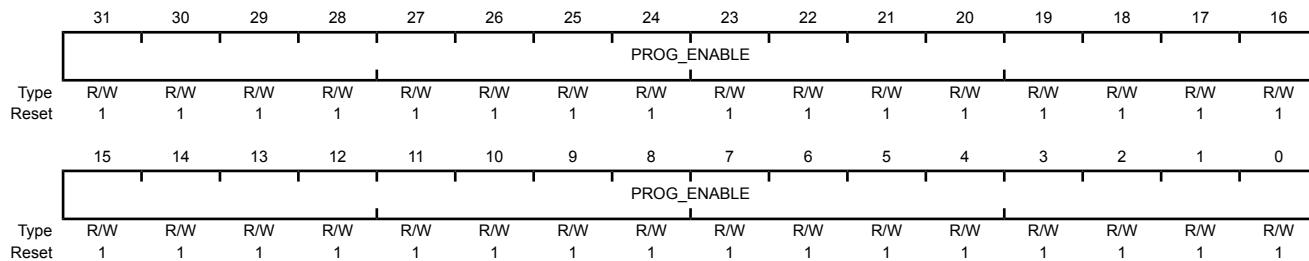
This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. This register is loaded during the 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. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 64 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 1 (FMPPE1)

Base 0x400F.E000

Offset 0x404

Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash Programming Enable Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table "Flash Protection Policy Combinations".
				Value Description
				0xFFFFFFFF Bits [31:0] each enable protection on a 2-KB block of Flash memory in memory range from 65 to 128 KB.

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 (**FMPREn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. 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. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 128 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. 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															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PROG_ENABLE																
Type	R/W															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field Name Type Reset Description

31:0 PROG_ENABLE R/W 0x00000000 Flash Programming Enable
Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table "Flash Protection Policy Combinations".

Value Description

0x00000000 Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 129 to 192 KB.

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 (**FMPREn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. 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. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 3 (FMPPE3)

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

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

Bit/Field Name Type Reset Description

31:0 PROG_ENABLE R/W 0x00000000 Flash Programming Enable
Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table "Flash Protection Policy Combinations".

Value Description

0x00000000 Bits [31:0] each enable protection on a 2-KB block of Flash memory in the range from 193 to 256 KB.

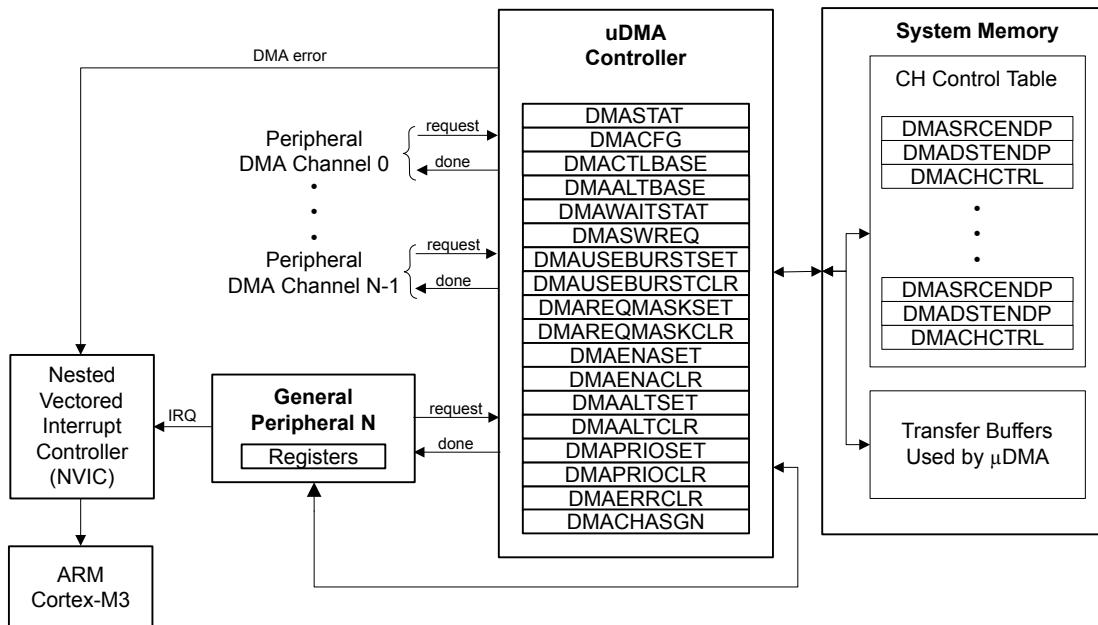
8 Micro Direct Memory Access (μ DMA)

The LM3S9L97 microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (μ DMA). The μ 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 available bus bandwidth. The μ 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 μ DMA controller provides the following features:

- ARM PrimeCell® 32-channel configurable μ DMA controller
- Support for memory-to-memory, memory-to-peripheral, and peripheral-to-memory in multiple transfer modes
 - 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
 - Primary and secondary 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 μ DMA controller and the processor core
 - μ 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 peripheral requests

8.1 Block Diagram

Figure 8-1. μ DMA Block Diagram



8.2 Functional Description

The μ 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 μ DMA controller's usage of the bus is always subordinate to the processor core, so it never holds up a bus transaction by the processor. Because the μ 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 enhance the ability of the processor core and the μ DMA controller to efficiently share the on-chip bus, thus improving performance. The optimizations include RAM striping and peripheral bus segmentation, which in many cases allow both the processor core and the μ DMA controller to access the bus and perform simultaneous data transfers.

The μ DMA controller can transfer data to and from the on-chip SRAM. However, because the Flash memory and ROM are located on a separate internal bus, it is not possible to transfer data from the Flash memory or ROM with the μ DMA controller.

Each peripheral function that is supported has a dedicated channel on the μ DMA controller that can be configured independently. The μ 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 μ DMA controller to perform arbitrary-sized transfers to and from arbitrary locations as part of a single transfer request. The μ 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 μ DMA controller rearbitrates 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 µDMA service request.

8.2.1 Channel Assignments

µDMA channels 0-31 are assigned to peripherals according to the following table. The **DMA Channel Assignment (DMACHASGN)** register (see page 413) can be used to specify the primary or secondary assignment. If the primary function is not available on this microcontroller, the secondary function becomes the primary function. If the secondary function is not available, the primary function is the only option.

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.

The USB endpoints mapped to µDMA channels 0-3 can be changed with the **USBDMASEL** register (see page 1051).

If a channel is marked with "*" below and is configured to transfer data with a software request using the **DMASWREQ** register, this channel must also be enabled in the **DMAENASET** register.

Table 8-1. µDMA Channel Assignments

µDMA Channel	Primary Assignment	Secondary 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	Ethernet Receive	General-Purpose Timer 2A*
7	Ethernet Transmit	General-Purpose Timer 2B*
8	UART0 Receive	UART1 Receive
9	UART0 Transmit	UART1 Transmit
10	SSI0 Receive	SSI1 Receive
11	SSI0 Transmit	SSI1 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	Available for software
21	General-Purpose Timer 1B	Available for software
22	UART1 Receive	Available for software
23	UART1 Transmit	Available for software
24	SSI1 Receive	ADC1 Sample Sequencer 0*

Table 8-1. μ DMA Channel Assignments (continued)

μ DMA Channel	Primary Assignment	Secondary Assignment
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 ² S0 Receive	Available for software
29	I ² S0 Transmit	Available for software
30	Dedicated for software use	
31	Reserved	

8.2.2 Priority

The μ 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 μ DMA channel requests a transfer, the μ DMA controller arbitrates among all the channels making a request and services the μ DMA channel with the highest priority. Once a transfer begins, it continues for a selectable number of transfers before rearbitrating among the requesting channels again. The arbitration size can be configured for each channel, ranging from 1 to 1024 item transfers. After the μ 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 μ DMA channel uses a large arbitration size, the latency for higher priority channels is increased because the μ 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 μ DMA channel priority, not arbitration for the bus. When the μ DMA controller arbitrates for the bus, the processor always takes priority. Furthermore, the μ 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 μ 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 μ DMA controller responds differently depending on whether the peripheral is making a single request or a burst request. If both are asserted, and the μ DMA channel has been set up for a burst transfer, then the burst request takes precedence. See Table 8-2 on page 369, which shows how each peripheral supports the two request types.

Table 8-2. Request Type Support

Peripheral	Single Request Signal	Burst Request Signal
ADC	None	Sequencer IE bit
Ethernet TX	TX FIFO empty	None
Ethernet RX	RX packet received	None
General-Purpose Timer	Raw interrupt pulse	None
I ² S TX	None	FIFO service request
I ² S RX	None	FIFO service request
SSI TX	TX FIFO Not Full	TX FIFO Level (fixed at 4)
SSI RX	RX FIFO Not Empty	RX FIFO Level (fixed at 4)
UART TX	TX FIFO Not Full	TX FIFO Level (configurable)
UART RX	RX FIFO Not Empty	RX FIFO Level (configurable)
USB TX	None	FIFO TXRDY
USB RX	None	FIFO RXRDY

8.2.4.1 Single Request

When a single request is detected, and not a burst request, the µDMA 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 µDMA 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. A burst transfer runs to completion once it is started, and cannot be interrupted, even by a higher priority channel. Burst transfers complete in a shorter time than the same number of non-burst transfers.

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 µDMA controller only responds to burst requests for that channel.

8.2.5 Channel Configuration

The µDMA 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 µDMA 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 370 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
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 is aligned on a 16-byte boundary. The entry contains four long words: the source end pointer, the destination end pointer, the control word, and an unused entry. The end 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 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 “ μ DMA Channel Control Structure” on page 387. The μ 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 indicates “stopped.” Because the control word is modified by the μ 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 μDMA 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 μDMA transfer, the controller automatically disables the channel.

8.2.6 Transfer Modes

The μDMA 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 μDMA controller does not perform any transfers and disables the channel if it is enabled. At the end of a transfer, the μDMA controller updates the control word to set the mode to Stop.

8.2.6.2 Basic Mode

In Basic mode, the μDMA 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 μDMA 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 the number of transfers specified by the ARBSIZE field in the **DMA Channel Control Word (DMACHCTL)** register is transferred on a software request, even if there is more data to transfer.

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

8.2.6.3 Auto Mode

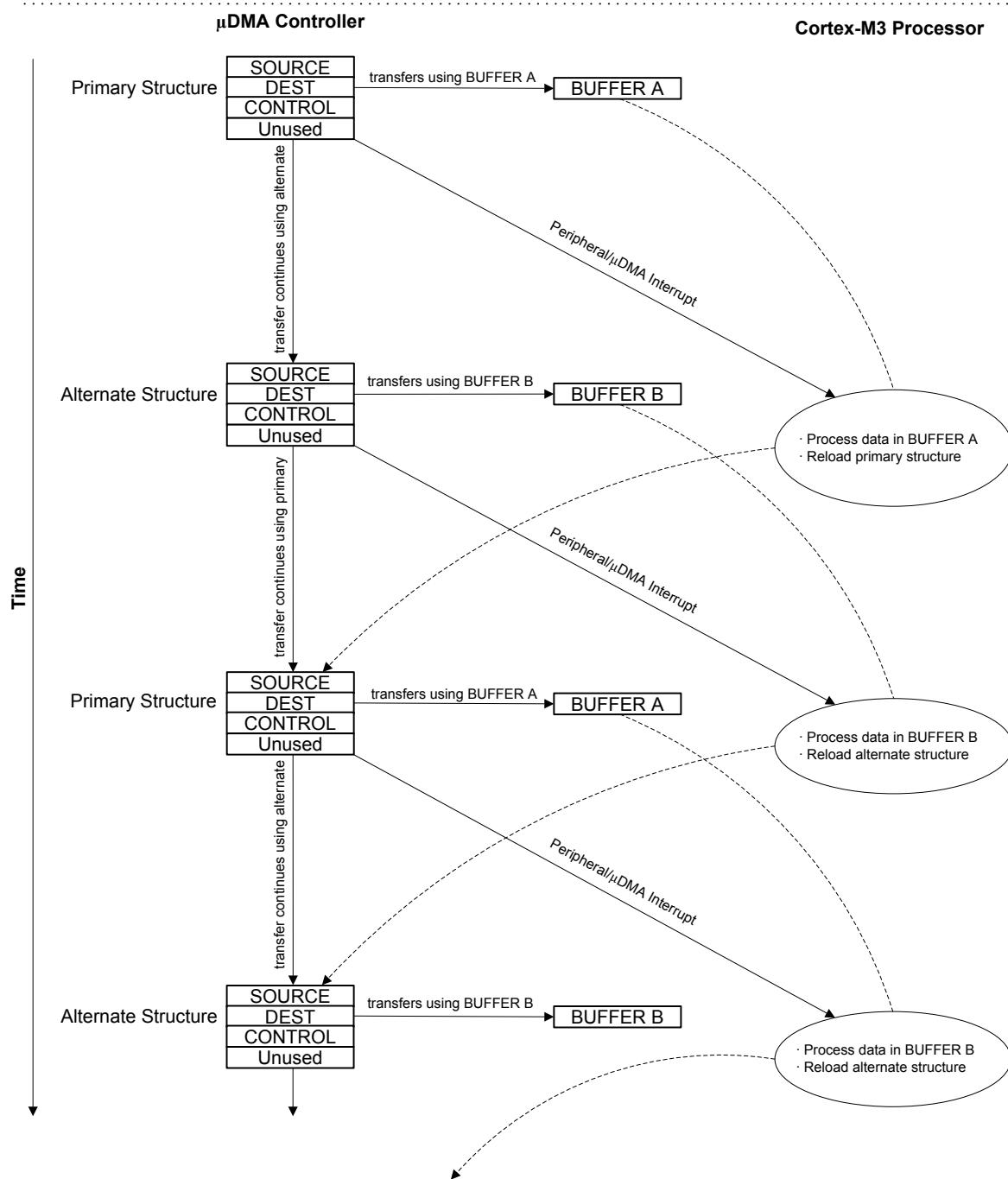
Auto mode is similar to Basic mode, except that once a transfer request is received, the transfer runs to completion, even if the μDMA request is removed. This mode is suitable for software-triggered transfers. Generally, Auto mode is not used with a peripheral.

When all the items have been transferred using Auto mode, the μDMA 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 μDMA 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 on page 372 for an example showing operation in Ping-Pong mode.

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

8.2.6.5 Memory Scatter-Gather

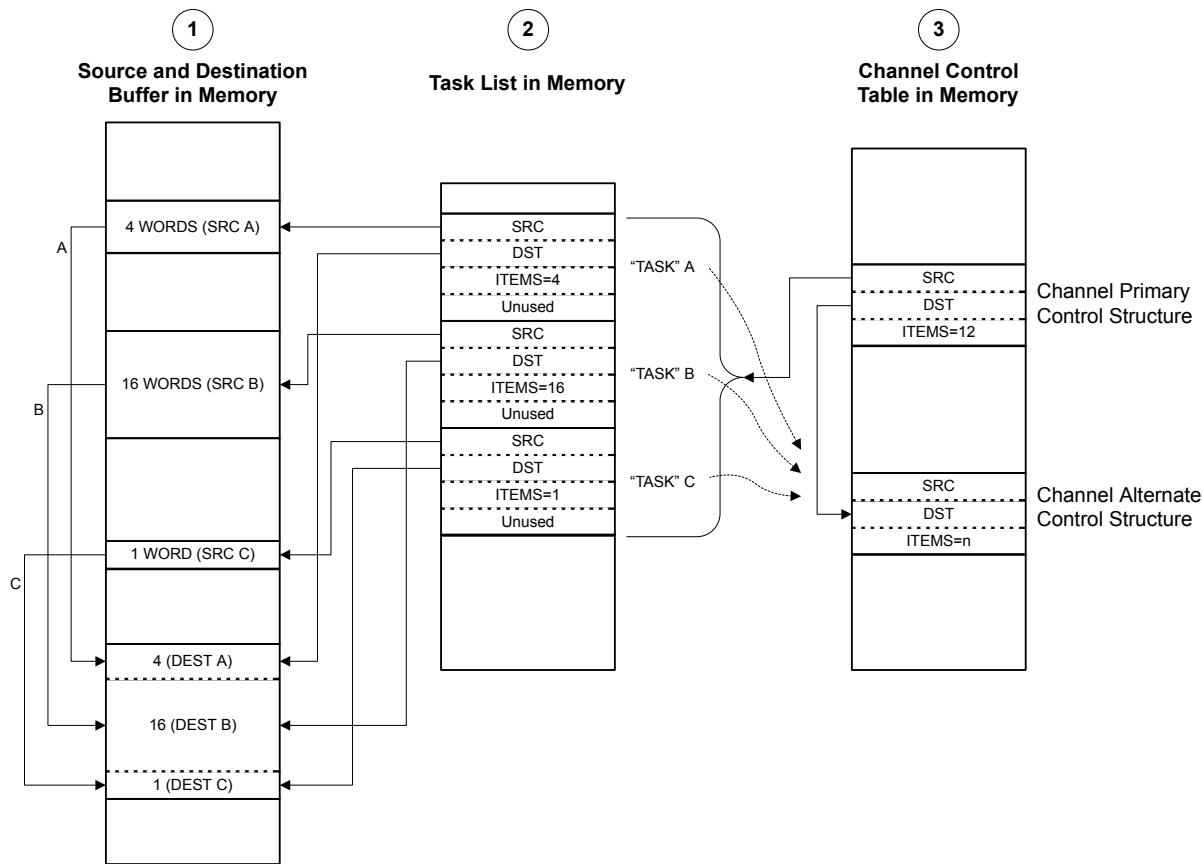
Memory Scatter-Gather mode is a complex mode used when data must be transferred to or from varied locations in memory instead of a set of contiguous locations in a memory buffer. For example, a gather μ DMA operation could be used to selectively read the payload of several stored packets of a communication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

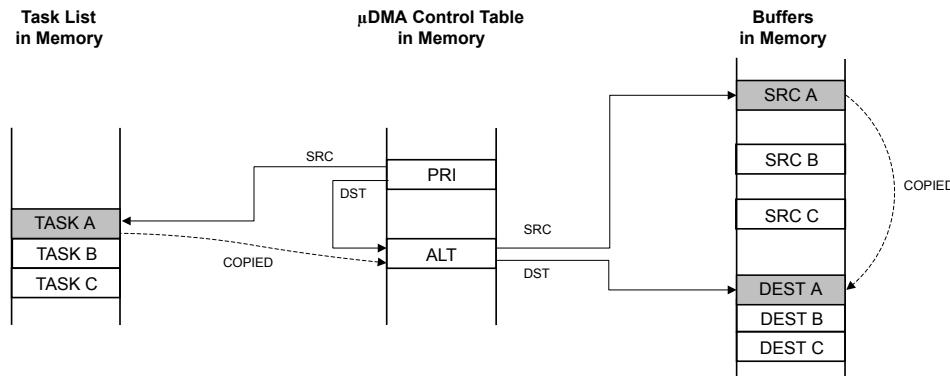
By programming the μDMA controller using this method, a set of arbitrary transfers can be performed based on a single μDMA request.

Refer to Figure 8-3 on page 374 and Figure 8-4 on page 375, 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 374 shows how the application sets up a μDMA 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 375 shows the sequence as the μDMA controller performs the three sets of copy operations. First, using the primary control structure, the μDMA 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 μDMA 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.

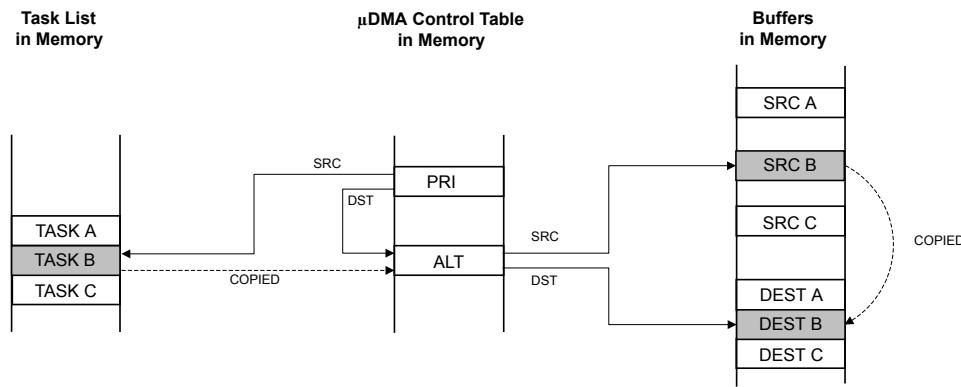
Figure 8-3. Memory Scatter-Gather, Setup and Configuration**NOTES:**

1. Application has a need to copy data items from three separate locations in memory into one combined buffer.
2. Application sets up μ DMA “task list” in memory, which contains the pointers and control configuration for three μ 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 is executed by the μ DMA controller.

Figure 8-4. Memory Scatter-Gather, μDMA Copy Sequence

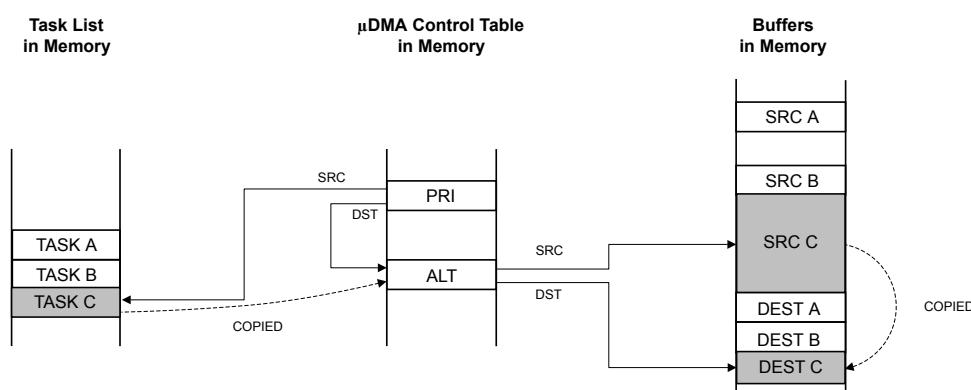
Using the channel's primary control structure, the μDMA controller copies task A configuration to the channel's alternate control structure.

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



Using the channel's primary control structure, the μDMA controller copies task B configuration to the channel's alternate control structure.

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



Using the channel's primary control structure, the μDMA controller copies task C configuration to the channel's alternate control structure.

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

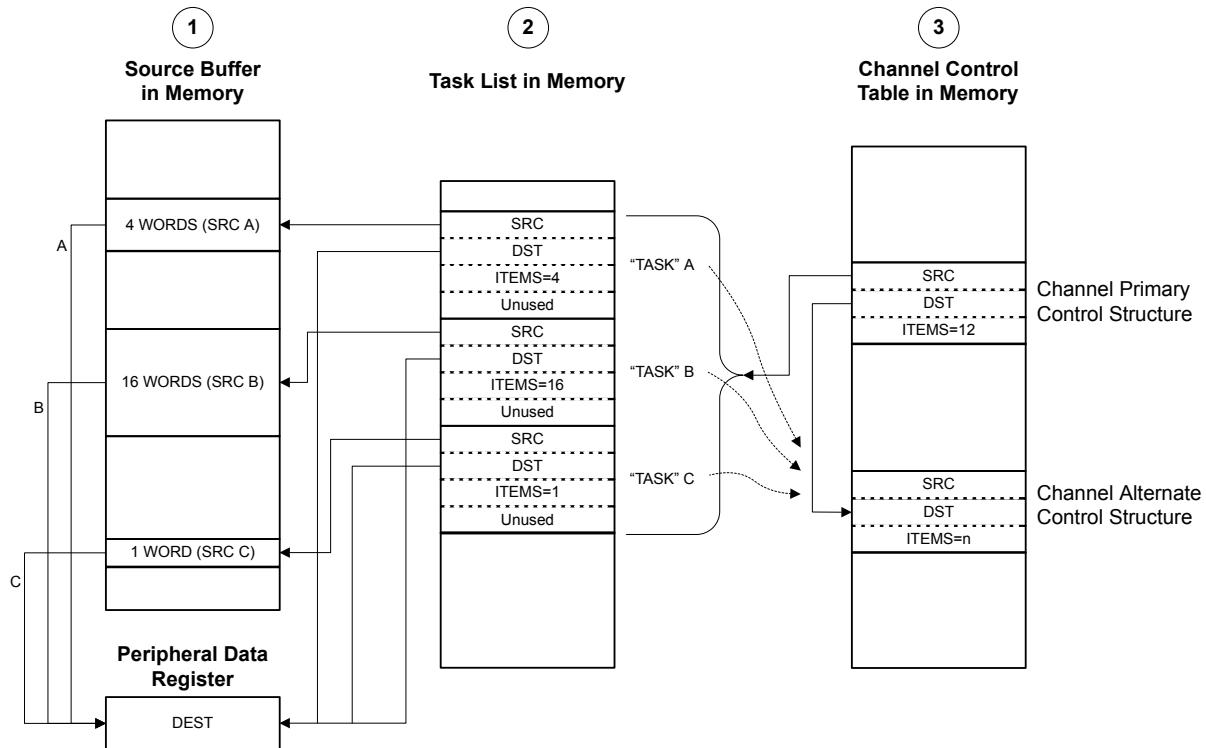
8.2.6.6 Peripheral Scatter-Gather

Peripheral Scatter-Gather mode is very similar to Memory Scatter-Gather, except that the transfers are controlled by a peripheral making a μ DMA request. Upon detecting a request from the peripheral, the μ 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 μ DMA request. The μ 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.

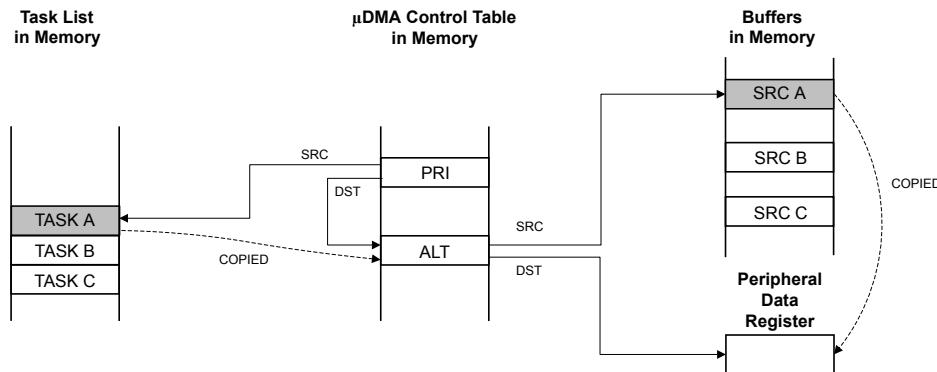
By using this method, the μ DMA controller can transfer data to or from a peripheral from a set of arbitrary locations whenever the peripheral is ready to transfer data.

Refer to Figure 8-5 on page 377 and Figure 8-6 on page 378, 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 377 shows how the application sets up a μ 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 378 shows the sequence as the μ DMA controller performs the three sets of copy operations. First, using the primary control structure, the μ 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 μ 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.

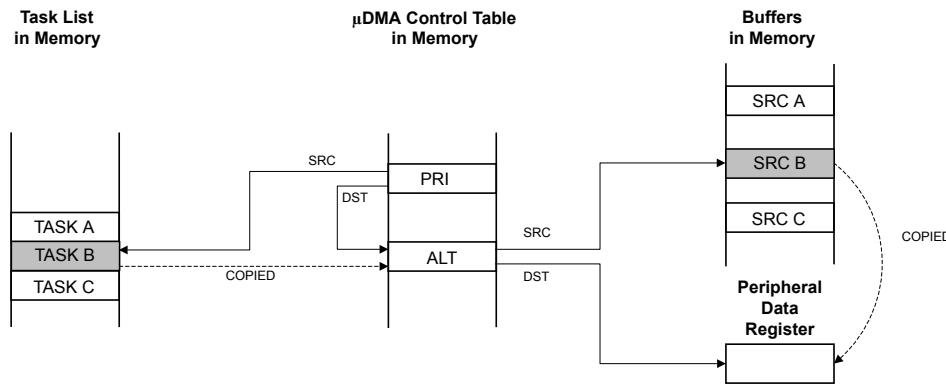
Figure 8-5. Peripheral Scatter-Gather, Setup and Configuration**NOTES:**

1. Application has a need to copy data items from three separate locations in memory into a peripheral data register.
2. Application sets up μDMA “task list” in memory, which contains the pointers and control configuration for three μDMA 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 is executed by the μDMA controller.

Figure 8-6. Peripheral Scatter-Gather, μ DMA Copy Sequence

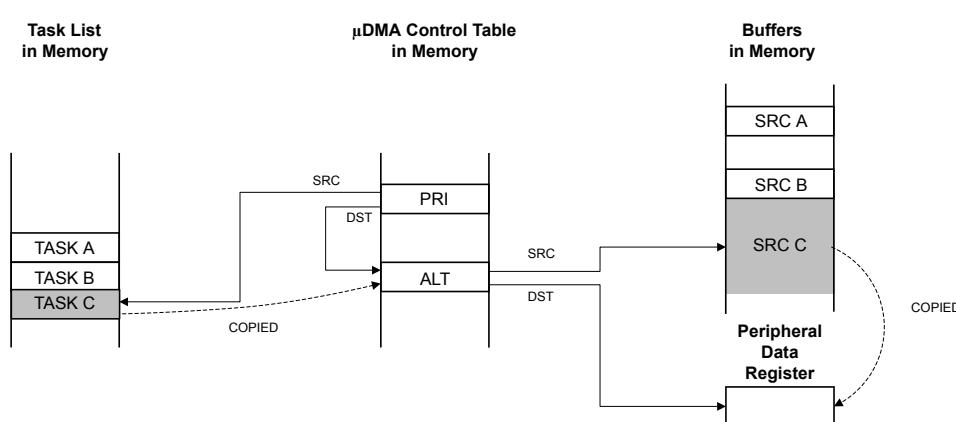
Using the channel's primary control structure, the μ DMA controller copies task A configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the μ DMA controller copies data from the source buffer A to the peripheral data register.



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

Then, using the channel's alternate control structure, the μ DMA controller copies data from the source buffer B to the peripheral data register.



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

Then, using the channel's alternate control structure, the μ DMA controller copies data from the source buffer C to the peripheral data register.

8.2.7 Transfer Size and Increment

The μDMA 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. μDMA 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 μDMA has a single request and/or burst request signal that is asserted when the peripheral is ready to transfer data (see Table 8-2 on page 369). 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 μDMA request signal is disabled, or masked, when the channel request mask bit is set. When the request is not masked, the μDMA channel is configured correctly and enabled, and the peripheral asserts the request signal, the μDMA controller begins the transfer.

When a μDMA transfer is complete, the μDMA controller generates an interrupt, see “Interrupts and Errors” on page 380 for more information.

For more information on how a specific peripheral interacts with the μDMA controller, refer to the DMA Operation section in the chapter that discusses that peripheral.

8.2.9 Software Request

One μDMA channel is dedicated to software-initiated transfers. This channel also has a dedicated interrupt to signal completion of a μDMA 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 μDMA 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 μDMA for data transfer.

8.2.10 Interrupts and Errors

When a μ DMA transfer is complete, the μ DMA controller generates a completion interrupt on the interrupt vector of the peripheral. Therefore, if μ 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 μ DMA transfer completion interrupt. If the transfer uses the software μ DMA channel, then the completion interrupt occurs on the dedicated software μ DMA interrupt vector (see Table 8-6 on page 380).

When μ DMA is enabled for a peripheral, the μ DMA controller stops the normal transfer interrupts for a peripheral from reaching the interrupt controller (the interrupts are still reported in the peripheral's interrupt registers). Thus, when a large amount of data is transferred using μ DMA, instead of receiving multiple interrupts from the peripheral as data flows, the interrupt controller receives only one interrupt when the transfer is complete. Unmasked peripheral error interrupts continue to be sent to the interrupt controller.

If the μ DMA controller encounters a bus or memory protection error as it attempts to perform a data transfer, it disables the μ DMA channel that caused the error and generates an interrupt on the μ 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.

Table 8-6 shows the dedicated interrupt assignments for the μ DMA controller.

Table 8-6. μ DMA Interrupt Assignments

Interrupt	Assignment
46	μ DMA Software Channel Transfer
47	μ DMA Error

8.3 Initialization and Configuration

8.3.1 Module Initialization

Before the μ 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 μ DMA peripheral must be enabled in the System Control block. To do this, set the `UDMA` bit of the System Control **RCGC2** register (see page 285).
2. Enable the μ 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

μ 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 μDMA controller to respond to single and burst requests.
4. Set bit 30 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the μDMA 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 μ 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 μ 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 μ 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 μ 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 μ 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 μ 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
DSTSIZEx	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 μDMA controller is now configured for transfer on channel 7. The controller makes transfers to the peripheral whenever the peripheral asserts a μDMA request. The transfers continue until the entire buffer of 64 bytes has been transferred. When that happens, the μDMA 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 μDMA 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 μDMA 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 μ 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 μ 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 μ 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 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
DSTSIZEx	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 μDMA 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 μDMA 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 μDMA controller is now configured and enabled for transfer on channel 8. When the peripheral asserts the μDMA request signal, the μDMA 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 385.
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 385.

8.3.5 Configuring Channel Assignments

Channel assignments for each μ DMA channel can be changed using the **DMACHASGN** register. Each bit represents a μ DMA channel. If the bit is set, then the secondary function is used for the channel.

Refer to Table 8-1 on page 367 for channel assignments.

For example, to use SSI1 Receive on channel 8 instead of UART0, set bit 8 of the **DMACHASGN** register.

8.4 Register Map

Table 8-13 on page 386 lists the μ 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 369 and Table 8-3 on page 370 for a description of how the entries in the channel control table are located in memory. The μ DMA register addresses are given as a hexadecimal increment, relative to the μ DMA base address of 0x400F.F000. Note that the μ DMA module clock must be enabled before the registers can be programmed (see page 285).

Table 8-13. μ DMA Register Map

Offset	Name	Type	Reset	Description	See page
μDMA Channel Control Structure (Offset from Channel Control Table Base)					
0x000	DMASRCENDP	R/W	-	DMA Channel Source Address End Pointer	388
0x004	DMADSTENDP	R/W	-	DMA Channel Destination Address End Pointer	389
0x008	DMACHCTL	R/W	-	DMA Channel Control Word	390
μDMA Registers (Offset from μDMA Base Address)					
0x000	DMASTAT	RO	0x001F.0000	DMA Status	395
0x004	DMACFG	WO	-	DMA Configuration	397
0x008	DMACTLBASE	R/W	0x0000.0000	DMA Channel Control Base Pointer	398

Table 8-13. μDMA Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x00C	DMAALTBASE	RO	0x0000.0200	DMA Alternate Channel Control Base Pointer	399
0x010	DMAWAITSTAT	RO	0x0000.0000	DMA Channel Wait-on-Request Status	400
0x014	DMASWREQ	WO	-	DMA Channel Software Request	401
0x018	DMAUSEBURSTSET	R/W	0x0000.0000	DMA Channel Useburst Set	402
0x01C	DMAUSEBURSTCLR	WO	-	DMA Channel Useburst Clear	403
0x020	DMAREQMASKSET	R/W	0x0000.0000	DMA Channel Request Mask Set	404
0x024	DMAREQMASKCLR	WO	-	DMA Channel Request Mask Clear	405
0x028	DMAENASET	R/W	0x0000.0000	DMA Channel Enable Set	406
0x02C	DMAENACLR	WO	-	DMA Channel Enable Clear	407
0x030	DMAALTSET	R/W	0x0000.0000	DMA Channel Primary Alternate Set	408
0x034	DMAALTCLR	WO	-	DMA Channel Primary Alternate Clear	409
0x038	DMAPRIOSET	R/W	0x0000.0000	DMA Channel Priority Set	410
0x03C	DMAPRIOCLR	WO	-	DMA Channel Priority Clear	411
0x04C	DMAERRCLR	R/W	0x0000.0000	DMA Bus Error Clear	412
0x500	DMACHASGN	R/W	0x0000.0000	DMA Channel Assignment	413
0xFD0	DMAPeriphID4	RO	0x0000.0004	DMA Peripheral Identification 4	418
0xFE0	DMAPeriphID0	RO	0x0000.0030	DMA Peripheral Identification 0	414
0xFE4	DMAPeriphID1	RO	0x0000.00B2	DMA Peripheral Identification 1	415
0xFE8	DMAPeriphID2	RO	0x0000.000B	DMA Peripheral Identification 2	416
0xFEC	DMAPeriphID3	RO	0x0000.0000	DMA Peripheral Identification 3	417
0xFF0	DMAPCellID0	RO	0x0000.000D	DMA PrimeCell Identification 0	419
0xFF4	DMAPCellID1	RO	0x0000.00F0	DMA PrimeCell Identification 1	420
0xFF8	DMAPCellID2	RO	0x0000.0005	DMA PrimeCell Identification 2	421
0xFFC	DMAPCellID3	RO	0x0000.00B1	DMA PrimeCell Identification 3	422

8.5 μDMA Channel Control Structure

The μDMA Channel Control Structure holds the transfer settings for a μDMA channel. Each channel has two control structures, which are located in a table in system memory. Refer to “Channel Configuration” on page 369 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

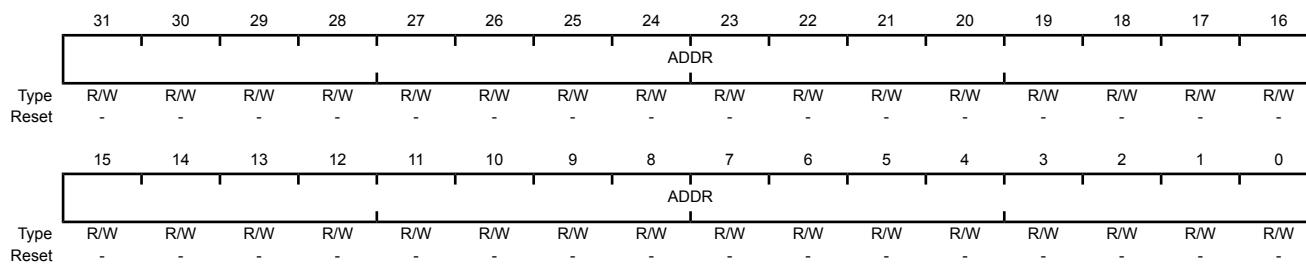
DMA Channel Source Address End Pointer (DMASRCENDP) is part of the Channel Control Structure and is used to specify the source address for a μ DMA transfer.

The μ DMA controller can transfer data to and from the on-chip SRAM. However, because the Flash memory and ROM are located on a separate internal bus, it is not possible to transfer data from the Flash memory or ROM with the μ DMA controller.

Note: The offset specified is from the base address of the control structure in system memory, not the μ 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 μ 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).

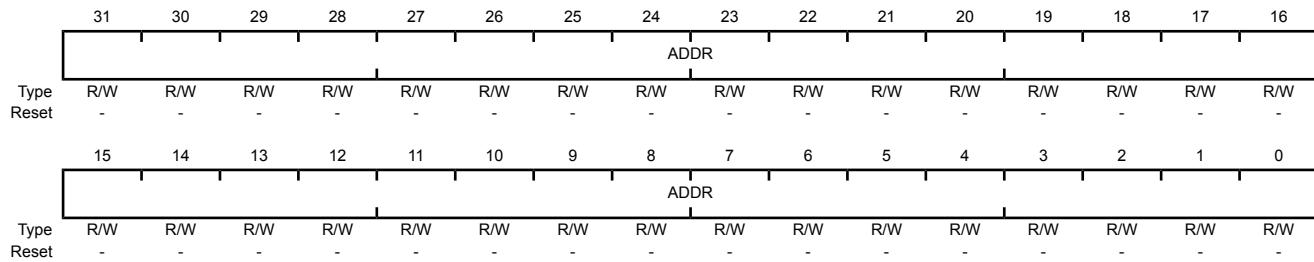
Register 2: DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004

DMA Channel Destination Address End Pointer (DMADSTENDP) is part of the Channel Control Structure and is used to specify the destination address for a µDMA transfer.

Note: The offset specified is from the base address of the control structure in system memory, not the µDMA module base address.

DMA Channel Destination Address End Pointer (DMADSTENDP)

Base n/a
Offset 0x004
Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:0	ADDR	R/W	-	Destination Address End Pointer This field points to the last address of the µDMA 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 μ DMA transfer.

Note: The offset specified is from the base address of the control structure in system memory, not the μ DMA module base address.

DMA Channel Control Word (DMACHCTL)

Base n/a
Offset 0x008
Type R/W, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DSTINC		DSTSIZE		SRCINC		SRCSIZE		reserved							ARBSIZE
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
	ARBSIZE						XFERSIZE						NXTUSEBURST		XFERMODE	
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										
31:30	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> <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													
29:28	DSTSIZE	R/W	-	<p>Destination Data Size</p> <p>This field configures the destination item data size.</p> <p>Note: DSTSIZE must be the same as SRCSIZE.</p> <table> <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										
27:26	SRCINC	R/W	-	<p>Source Address Increment This field configures the source address increment. The address increment value must be equal or greater than the value of the source size (SRCSIZE).</p> <table> <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 (DMASRCENDP) 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 (DMASRCENDP) 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 (DMASRCENDP) for the channel													
25:24	SRCSIZE	R/W	-	<p>Source Data Size This field configures the source item data size.</p> <p>Note: DSTSIZE must be the same as SRCSIZE.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Byte 8-bit data size.</td></tr> <tr> <td>0x1</td><td>Half-word 16-bit data size.</td></tr> <tr> <td>0x2</td><td>Word 32-bit data size.</td></tr> <tr> <td>0x3</td><td>Reserved</td></tr> </tbody> </table>	Value	Description	0x0	Byte 8-bit data size.	0x1	Half-word 16-bit data size.	0x2	Word 32-bit data size.	0x3	Reserved
Value	Description													
0x0	Byte 8-bit data size.													
0x1	Half-word 16-bit data size.													
0x2	Word 32-bit data size.													
0x3	Reserved													
23:18	reserved	R/W	-	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										

Bit/Field	Name	Type	Reset	Description																								
17:14	ARBSIZE	R/W	-	<p>Arbitration Size</p> <p>This field configures the number of transfers that can occur before the μDMA controller re-arbitrates. The possible arbitration rate configurations represent powers of 2 and are shown below.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>1 Transfer Arbitrates after each μDMA 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 In this configuration, no arbitration occurs during the μDMA transfer because the maximum transfer size is 1024.</td></tr> </tbody> </table>	Value	Description	0x0	1 Transfer Arbitrates after each μ 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 In this configuration, no arbitration occurs during the μ DMA transfer because the maximum transfer size is 1024.
Value	Description																											
0x0	1 Transfer Arbitrates after each μ 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 In this configuration, no arbitration occurs during the μ DMA transfer because the maximum transfer size is 1024.																											
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 μDMA controller updates this field immediately prior to entering the arbitration process, so it contains the number of outstanding items that is necessary to complete the μDMA cycle.</p>																								
3	NXTUSEBURST	R/W	-	<p>Next Useburst</p> <p>This field controls whether the Useburst SET[n] 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 μDMA 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>μDMA Transfer Mode</p> <p>This field configures the operating mode of the μDMA cycle. Refer to “Transfer Modes” on page 371 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> <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 μDMA 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 μDMA 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 μDMA controller stops. An interrupt is generated on completion of the transfers configured by each control structure. See “Ping-Pong” on page 371.

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 μDMA 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 372.

Alternate Memory Scatter-Gather

This value must be used in the alternate channel control data structure when the μ DMA controller operates in Memory Scatter-Gather mode.

Peripheral Scatter-Gather

This value must be used in the primary channel control data structure when the μ DMA controller operates in Peripheral Scatter-Gather mode. In this mode, the μ DMA 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 μ DMA 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 376.

Alternate Peripheral Scatter-Gather

This value must be used in the alternate channel control data structure when the μ DMA controller operates in Peripheral Scatter-Gather mode.

8.6 μ DMA Register Descriptions

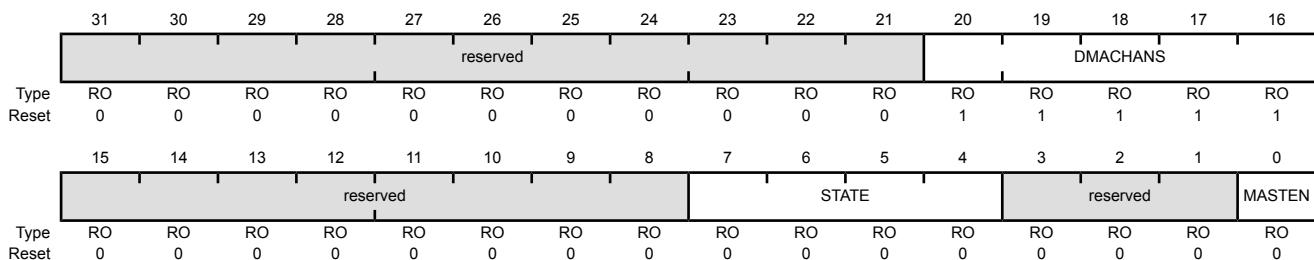
The register addresses given are relative to the μ DMA base address of 0x400F.F000.

Register 4: DMA Status (DMASTAT), offset 0x000

The **DMA Status (DMASTAT)** register returns the status of the µDMA controller. You cannot read this register when the µDMA controller is in the reset state.

DMA Status (DMASTAT)

Base 0x400F.F000
Offset 0x000
Type RO, reset 0x001F.0000



Bit/Field	Name	Type	Reset	Description
31:21	reserved	RO	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 µDMA Channels Minus 1 This field contains a value equal to the number of µDMA channels the µDMA controller is configured to use, minus one. The value of 0x1F corresponds to 32 µDMA 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.
7:4	STATE	RO	0x0	Control State Machine Status This field shows the current status of the control state machine. Status can be one of the following.
		Value	Description	
		0x0	Idle	
		0x1	Reading channel controller data.	
		0x2	Reading source end pointer.	
		0x3	Reading destination end pointer.	
		0x4	Reading source data.	
		0x5	Writing destination data.	
		0x6	Waiting for µDMA request to clear.	
		0x7	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.

Bit/Field	Name	Type	Reset	Description
0	MASTEN	RO	0	Master Enable Status
				Value Description
			0	The μ DMA controller is disabled.
			1	The μ DMA controller is enabled.

Register 5: DMA Configuration (DMACFG), offset 0x004

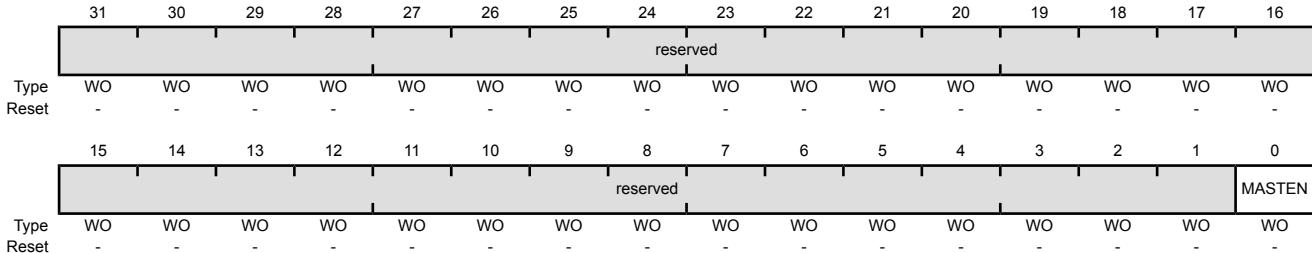
The **DMACFG** register controls the configuration of the µDMA controller.

DMA Configuration (DMACFG)

Base 0x400F.F000

Offset 0x004

Type WO, reset -



Bit/Field Name Type Reset Description

31:1 reserved WO - Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

0 MASTEN WO - Controller Master Enable

Value Description

0 Disables the µDMA controller.

1 Enables µDMA controller.

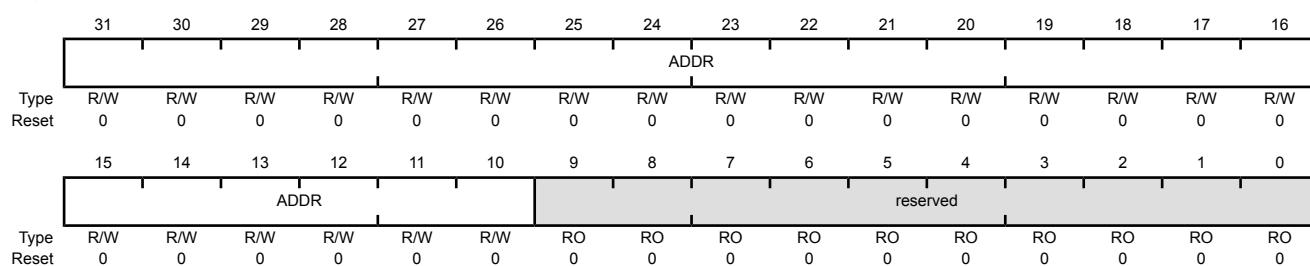
Register 6: DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008

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 μ DMA controller depends on the number of μ DMA channels used and whether the alternate channel control data structure is used. See “Channel Configuration” on page 369 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 μ 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 µDMA 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
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 μ DMA channel is waiting on a request. A peripheral can hold off the μ DMA from performing a single request until the peripheral is ready for a burst request to enhance the μ DMA performance. The use of this feature is dependent on the design of the peripheral and is not controllable by software in any way. This register cannot be read when the μ DMA controller is in the reset state.

DMA Channel Wait-on-Request Status (DMAWAITSTAT)

Base 0x400F.F000

Offset 0x010

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	WAITREQ[n]	RO	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

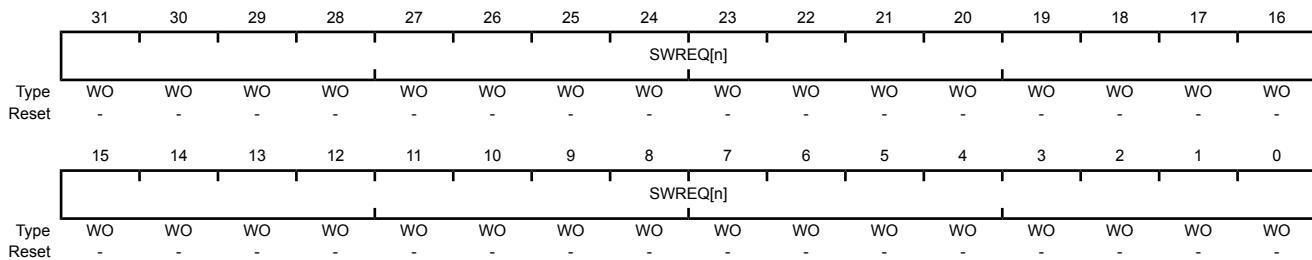
Each bit of the **DMASWREQ** register represents the corresponding µDMA channel. Setting a bit generates a request for the specified µDMA channel.

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.

These bits are automatically cleared when the software request has been completed.

Register 10: DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018

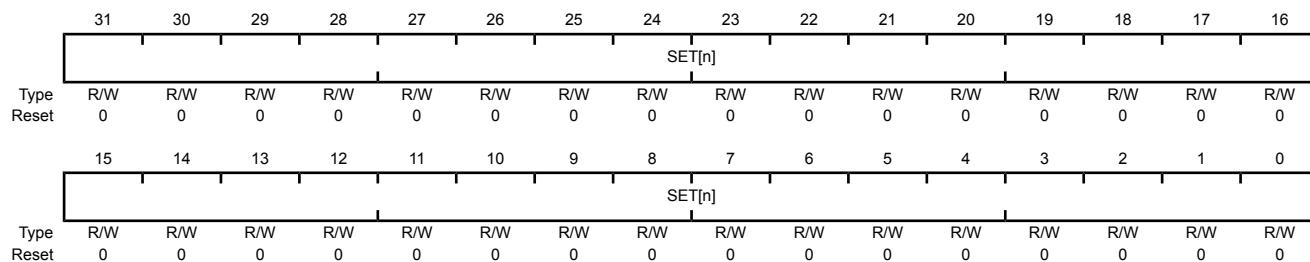
Each bit of the **DMAUSEBURSTSET** register represents the corresponding μ DMA channel. Setting a bit disables the channel's single request input from generating requests, configuring the channel to only accept burst requests. Reading the register returns the status of USEBURST.

If the amount of data to transfer is a multiple of the arbitration (burst) size, the corresponding SET[n] bit is cleared after completing the final transfer. If there are fewer items remaining to transfer than the arbitration (burst) size, the μ DMA controller automatically clears the corresponding SET[n] bit, allowing the remaining items to transfer using single requests. In order to resume transfers using burst requests, the corresponding bit must be set again. A bit should not be set if the corresponding peripheral does not support the burst request model.

Refer to “Request Types” on page 368 for more details about request types.

DMA Channel Useburst Set (DMAUSEBURSTSET)

Base 0x400F.F000
Offset 0x018
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R/W	0x0000.0000	Channel [n] Useburst Set
Value Description				
0 μ DMA channel [n] responds to single or burst requests.				
1 μ DMA channel [n] responds only to burst requests.				
Bit 0 corresponds to channel 0. This bit is automatically cleared as described above. A bit can also be manually cleared by setting the corresponding CLR[n] bit in the DMAUSEBURSTCLR register.				

Register 11: DMA Channel Useburst Clear (DMAUSEBURSTCLR), offset 0x01C

Each bit of the **DMAUSEBURSTCLR** register represents the corresponding µDMA channel. Setting a bit clears the corresponding `SET[n]` bit in the **DMAUSEBURSTSET** register.

DMA Channel Useburst Clear (DMAUSEBURSTCLR)

Base 0x400F.F000

Offset 0x01C

Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CLR[n]															
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
	CLR[n]															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Useburst Clear

Value Description

0 No effect.

1 Setting a bit clears the corresponding `SET[n]` bit in the **DMAUSEBURSTSET** register meaning that µDMA 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 μ DMA channel. Setting a bit disables μ DMA requests for the channel. Reading the register returns the request mask status. When a μ DMA channel's request is masked, that means the peripheral can no longer request μ DMA transfers. The channel can then be used for software-initiated transfers.

DMA Channel Request Mask Set (DMAREQMASKSET)

Base 0x400F.F000
Offset 0x020
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SET[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
	SET[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	SET[n]	R/W	0x0000.0000	Channel [n] Request Mask Set
------	--------	-----	-------------	------------------------------

Value	Description
0	The peripheral associated with channel [n] is enabled to request μ DMA transfers.
1	The peripheral associated with channel [n] is not able to request μ DMA transfers. Channel [n] may be used for software-initiated transfers.

Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding CLR[n] bit in the **DMAREQMASKCLR** register.

Register 13: DMA Channel Request Mask Clear (DMAREQMASKCLR), offset 0x024

Each bit of the **DMAREQMASKCLR** register represents the corresponding µDMA channel. Setting a bit clears the corresponding **SET[n]** bit in the **DMAREQMASKSET** register.

DMA Channel Request Mask Clear (DMAREQMASKCLR)

Base 0x400F.F000

Offset 0x024

Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CLR[n]															
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
	CLR[n]															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

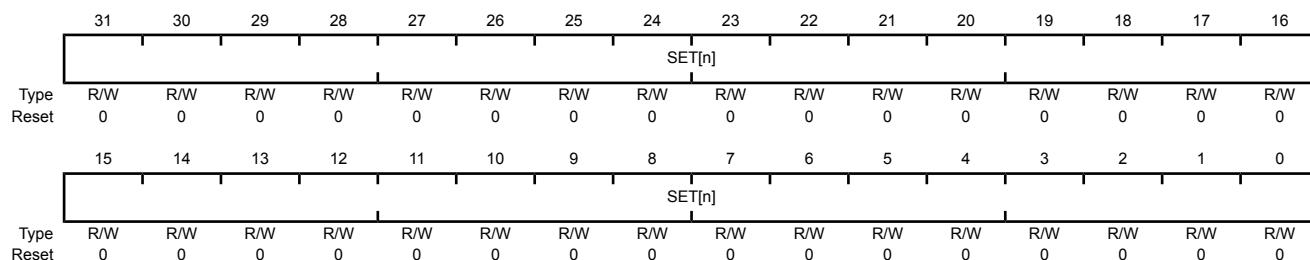
Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Request Mask Clear
Value Description				
0 No effect.				
1 Setting a bit clears the corresponding SET[n] bit in the DMAREQMASKSET register meaning that the peripheral associated with channel [n] is enabled to request µDMA transfers.				

Register 14: DMA Channel Enable Set (DMAENASET), offset 0x028

Each bit of the **DMAENASET** register represents the corresponding μ DMA channel. Setting a bit enables the corresponding μ 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.

DMA Channel Enable Set (DMAENASET)

Base 0x400F.F000
Offset 0x028
Type R/W, reset 0x0000.0000



Bit/Field Name Type Reset Description

31:0 SET[n] R/W 0x0000.0000 Channel [n] Enable Set

Value Description

- 0 μ DMA Channel [n] is disabled.
- 1 μ DMA Channel [n] is enabled.

Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding CLR[n] bit in the **DMAENACLR** register.

Register 15: DMA Channel Enable Clear (DMAENACLR), offset 0x02C

Each bit of the **DMAENACLR** register represents the corresponding µDMA channel. Setting a bit clears the corresponding **SET[n]** bit in the **DMAENASET** register.

DMA Channel Enable Clear (DMAENACLR)

Base 0x400F.F000

Offset 0x02C

Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CLR[n]															
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
	CLR[n]															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

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

31:0	CLR[n]	WO	-	Clear Channel [n] Enable Clear
------	--------	----	---	--------------------------------

Value Description

0 No effect.

1 Setting a bit clears the corresponding **SET[n]** bit in the **DMAENASET** register meaning that channel [n] is disabled for µDMA transfers.

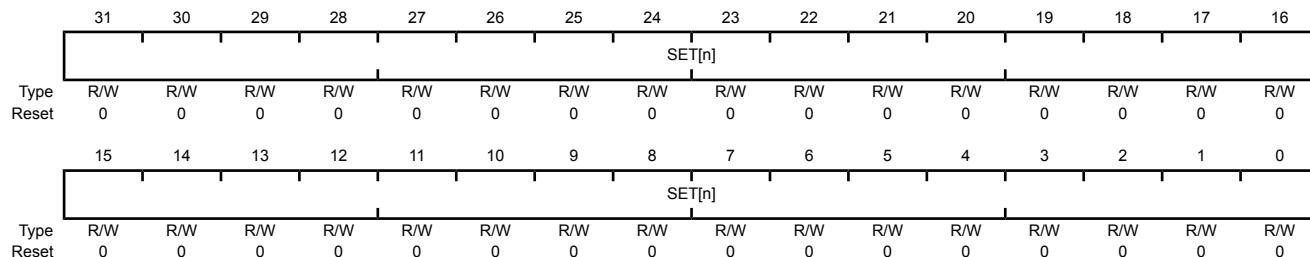
Note: The controller disables a channel when it completes the µDMA cycle.

Register 16: DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030

Each bit of the **DMAALTSET** register represents the corresponding μ DMA channel. Setting a bit configures the μ 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 μ DMA channel.

DMA Channel Primary Alternate Set (DMAALTSET)

Base 0x400F.F000
Offset 0x030
Type R/W, reset 0x0000.0000



Bit/Field Name Type Reset Description

31:0 SET[n] R/W 0x0000.0000 Channel [n] Alternate Set

Value Description

0 μ DMA channel [n] is using the primary control structure.

1 μ DMA channel [n] is using the alternate control structure.

Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding CLR[n] bit in the **DMAALTCLR** register.

Note: For Ping-Pong and Scatter-Gather cycle types, the μ DMA controller automatically sets these bits to select the alternate channel control data structure.

Register 17: DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034

Each bit of the **DMAALTCLR** register represents the corresponding µDMA channel. Setting a bit clears the corresponding `SET[n]` bit in the **DMAALTSET** register.

DMA Channel Primary Alternate Clear (DMAALTCLR)

Base 0x400F.F000

Offset 0x034

Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CLR[n]															
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
	CLR[n]															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Alternate Clear
Value Description				
0 No effect.				
1 Setting a bit clears the corresponding <code>SET[n]</code> bit in the DMAALTSET register meaning that channel [n] is using the primary control structure.				

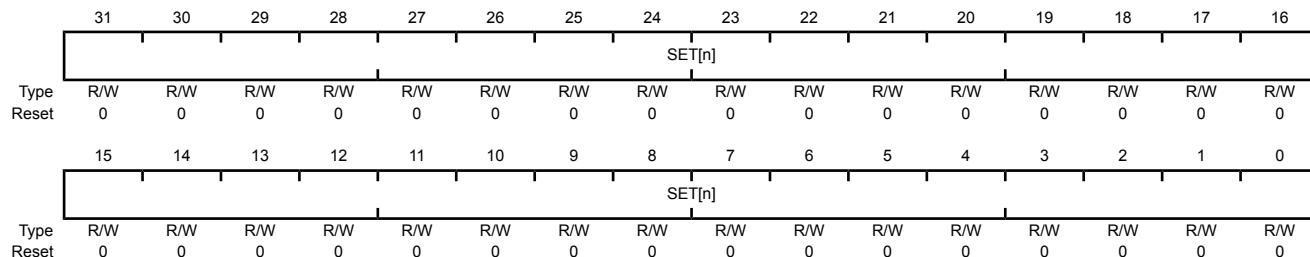
Note: For Ping-Pong and Scatter-Gather cycle types, the µDMA controller automatically sets these bits to select the alternate channel control data structure.

Register 18: DMA Channel Priority Set (DMAPRIOSET), offset 0x038

Each bit of the **DMAPRIOSET** register represents the corresponding μ DMA channel. Setting a bit configures the μ DMA channel to have a high priority level. Reading the register returns the status of the channel priority mask.

DMA Channel Priority Set (DMAPRIOSET)

Base 0x400F.F000
Offset 0x038
Type R/W, reset 0x0000.0000



Bit/Field Name Type Reset Description

31:0 SET[n] R/W 0x0000.0000 Channel [n] Priority Set

Value Description

0 μ DMA channel [n] is using the default priority level.

1 μ DMA channel [n] is using a high priority level.

Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding CLR[n] bit in the **DMAPRIOCLR** register.

Register 19: DMA Channel Priority Clear (DMAPRIOCLR), offset 0x03C

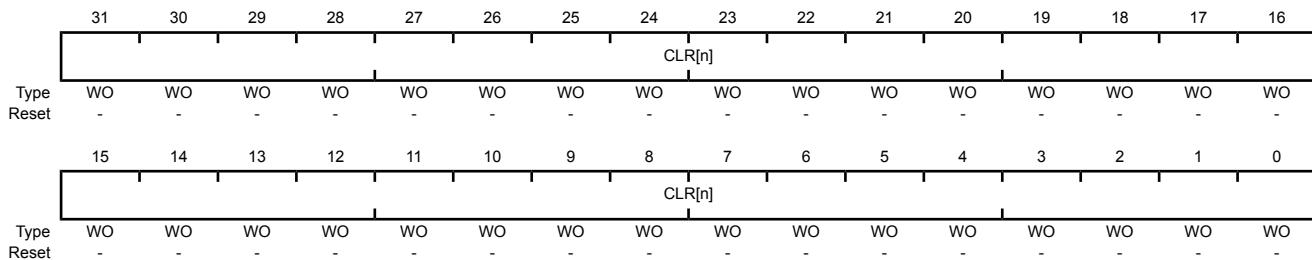
Each bit of the **DMAPRIOCLR** register represents the corresponding µDMA channel. Setting a bit clears the corresponding **SET[n]** bit in the **DMAPRIOSET** register.

DMA Channel Priority Clear (DMAPRIOCLR)

Base 0x400F.F000

Offset 0x03C

Type WO, reset -



Bit/Field Name Type Reset Description

31:0 CLR[n] WO - Channel [n] Priority Clear

Value Description

0 No effect.

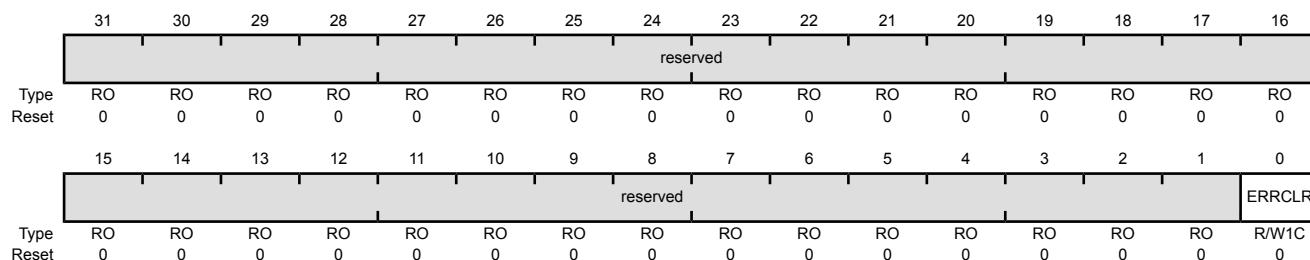
1 Setting a bit clears the corresponding **SET[n]** bit in the **DMAPRIOSET** register meaning that channel [n] is using the default priority level.

Register 20: DMA Bus Error Clear (DMAERRCLR), offset 0x04C

The **DMAERRCLR** register is used to read and clear the μ DMA bus error status. The error status is set if the μ DMA controller encountered a bus error while performing a transfer. If a bus error occurs on a channel, that channel is automatically disabled by the μ DMA controller. The other channels are unaffected.

DMA Bus Error Clear (DMAERRCLR)

Base 0x400F.F000
Offset 0x04C
Type R/W, reset 0x0000.0000



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

Value	Description
0	No bus error is pending.
1	A bus error is pending.

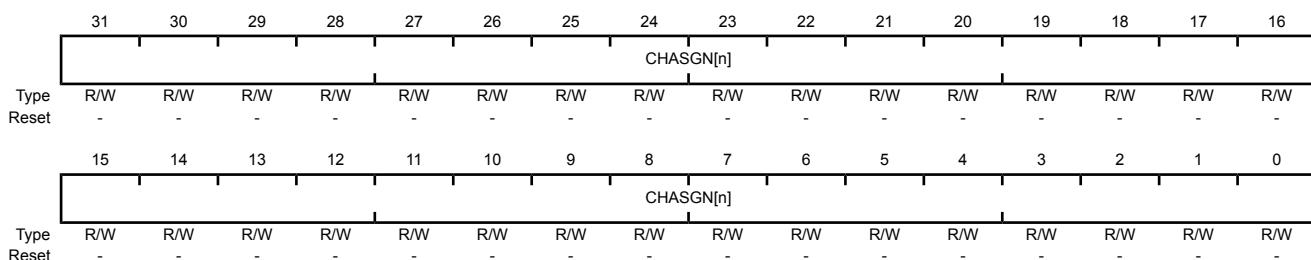
This bit is cleared by writing a 1 to it.

Register 21: DMA Channel Assignment (DMACHASGN), offset 0x500

Each bit of the **DMACHASGN** register represents the corresponding µDMA channel. Setting a bit selects the secondary channel assignment as specified in Table 8-1 on page 367.

DMA Channel Assignment (DMACHASGN)

Base 0x400F.F000
Offset 0x500
Type R/W, reset 0x0000.0000



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

31:0	CHASGN[n]	R/W	-	Channel [n] Assignment Select
------	-----------	-----	---	-------------------------------

Value Description

0 Use the primary channel assignment.

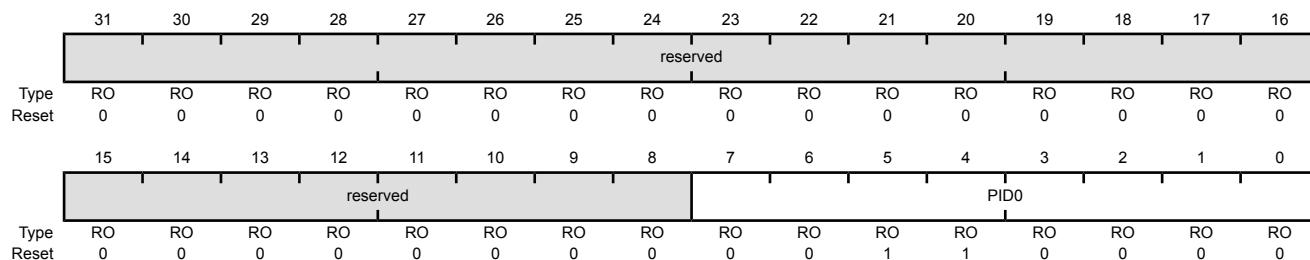
1 Use the secondary channel assignment.

Register 22: 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



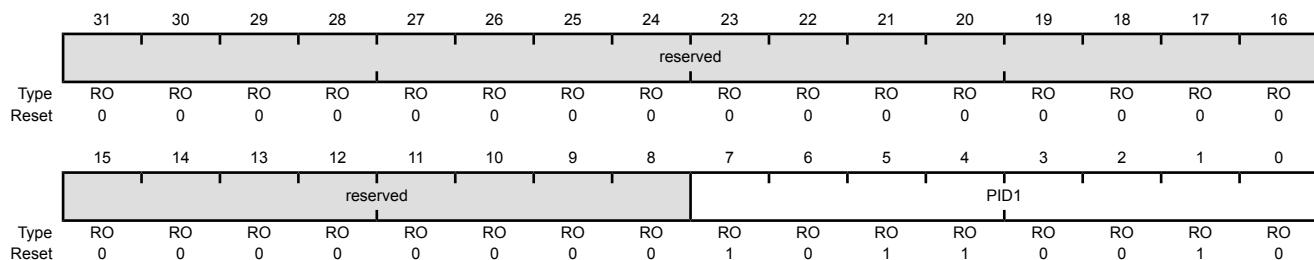
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x30	μ DMA Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

Register 23: 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



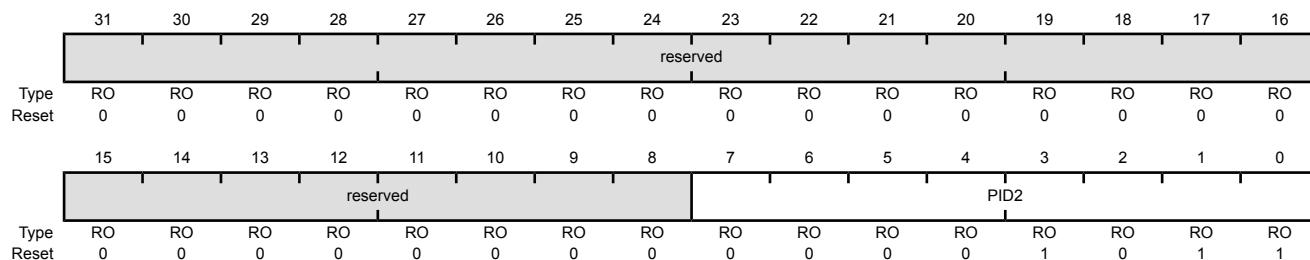
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0xB2	μDMA Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 24: DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8

The **DMAPeriphIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA Peripheral Identification 2 (DMAPeriphID2)

Base 0x400F.F000
Offset 0xFE8
Type RO, reset 0x0000.000B



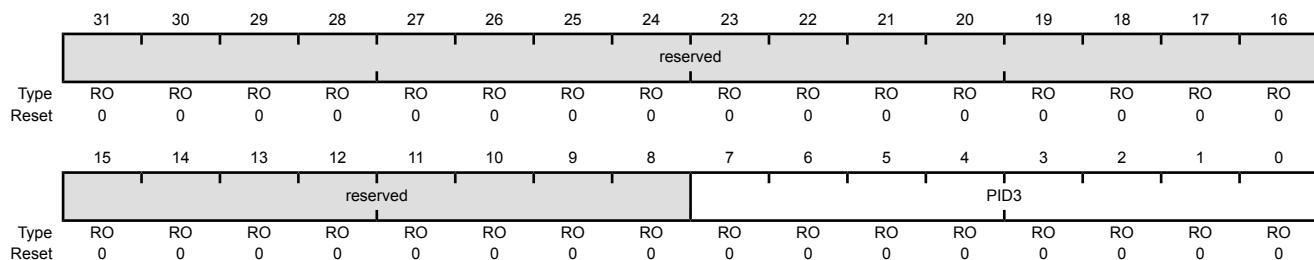
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x0B	μ DMA Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 25: 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



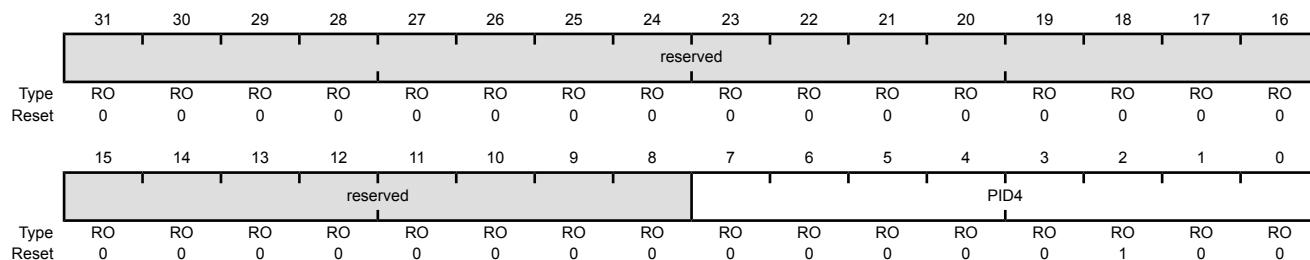
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x00	μDMA Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 26: 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



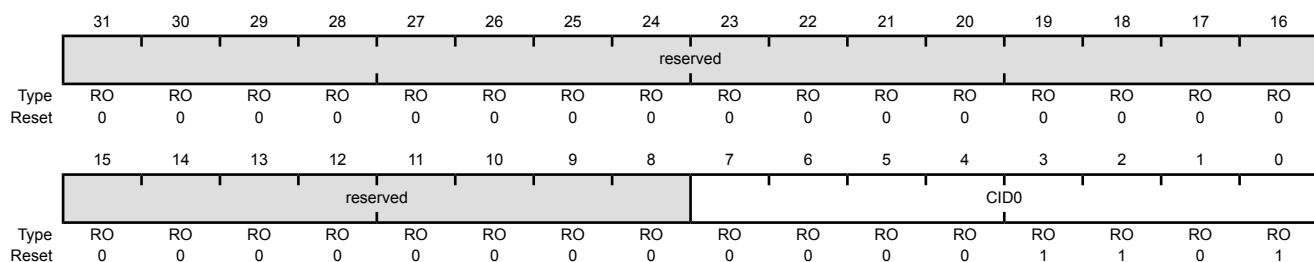
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x04	μ DMA Peripheral ID Register Can be used by software to identify the presence of this peripheral.

Register 27: DMA PrimeCell Identification 0 (DMACellID0), offset 0xFF0

The **DMACellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA PrimeCell Identification 0 (DMACellID0)

Base 0x400F.F000
Offset 0xFF0
Type RO, reset 0x0000.000D



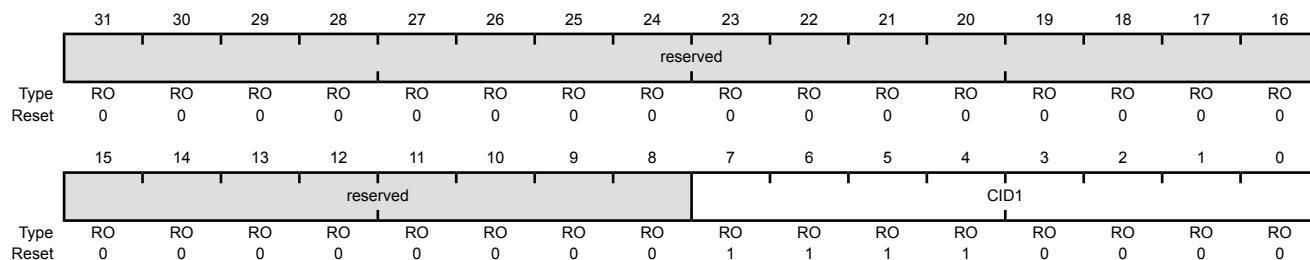
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	μDMA PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

Register 28: 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



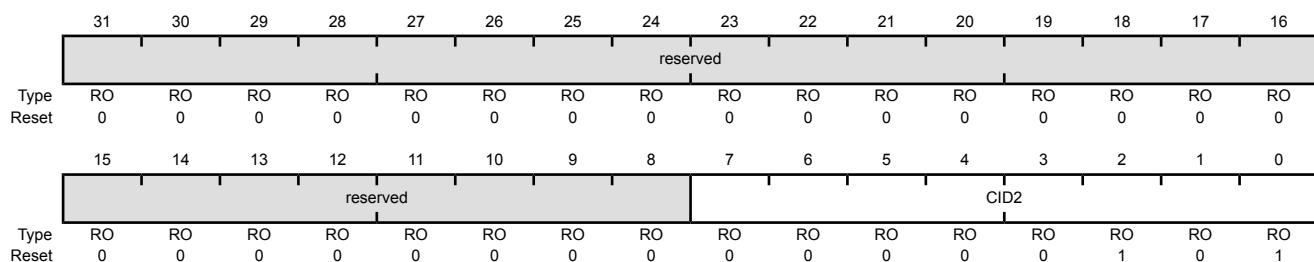
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	μ DMA PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

Register 29: DMA PrimeCell Identification 2 (DMACellID2), offset 0xFF8

The **DMACellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA PrimeCell Identification 2 (DMACellID2)

Base 0x400F.F000
Offset 0xFF8
Type RO, reset 0x0000.0005



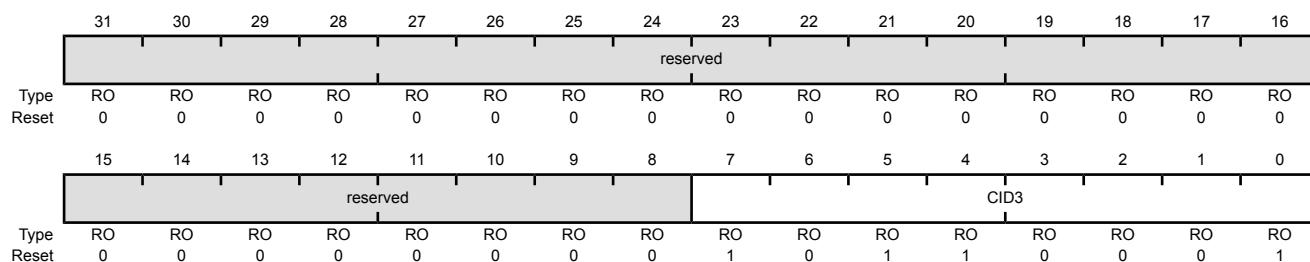
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	μDMA PrimeCell ID Register [23:16]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

Register 30: DMA PrimeCell Identification 3 (DMA_PCellID3), offset 0xFFC

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA PrimeCell Identification 3 (DMA_{PC}ID3)

Base 0x400F.F000
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	μDMA 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 up to 60 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

- Up to 60 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 5-V-tolerant in input configuration
- Fast toggle capable of a change every two clock cycles
- Two means of port access: either Advanced High-Performance 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 Signal Description

GPIO signals have alternate hardware functions. Table 9-2 on page 424 and Table 9-3 on page 426 list the GPIO pins and their analog and digital alternate functions. The **A_{IN}x** and **VREF_A** analog signals are not 5-V tolerant and go through an isolation circuit before reaching their circuitry. These signals are configured by clearing the corresponding **DEN** bit in the **GPIO Digital Enable (GPIODEN)** register and setting the corresponding **AMSEL** bit in the **GPIO Analog Mode Select (GPIOAMSEL)** register. Other analog signals are 5-V tolerant and are connected directly to their circuitry (**C₀-**,

C0+, C1-, C1+, USB0VBUS, USB0ID). These signals are configured by clearing the DEN bit in the **GPIO Digital Enable (GPIODEN)** register. The digital alternate hardware functions are enabled by setting the appropriate bit in the **GPIO Alternate Function Select (GPIOAFSEL)** and **GPIODEN** registers and configuring the PMC_x bit field in the **GPIO Port Control (GPIOPCTL)** register to the numeric encoding shown in the table below. Note that each pin must be programmed individually; no type of grouping is implied by the columns in the table. Table entries that are shaded gray are the default values for the corresponding GPIO pin.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL=0**, **GPIODEN=0**, **GPIOPDR=0**, **GPIOPUR=0**, and **GPIOPCTL=0**, with the exception of the four JTAG/SWD pins (shown in the table below). A Power-On-Reset (**POR**) or asserting **RST** puts the pins back to their default state.

Table 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 ² C0	1	1	0	0	0x1
PC[3:0]	JTAG/SWD	1	1	0	1	0x3

Table 9-2. GPIO Pins and Alternate Functions (100LQFP)

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMC _x Bit Field Encoding) ^a										
			1	2	3	4	5	6	7	8	9	10	11
PA0	26	-	U0Rx	-	-	-	-	-	-	I2C1SCL	U1Rx	-	-
PA1	27	-	U0Tx	-	-	-	-	-	-	I2C1SDA	U1Tx	-	-
PA2	28	-	SSI0Clk	-	-	PWM4	-	-	-	-	I2S0RXSD	-	-
PA3	29	-	SSI0Fss	-	-	PWM5	-	-	-	-	I2S0RXCLK	-	-
PA4	30	-	SSI0Rx	-	-	-	CAN0Rx	-	-	-	I2S0TXSCK	-	-
PA5	31	-	SSI0Tx	-	-	-	CAN0Tx	-	-	-	I2S0TXWS	-	-
PA6	34	-	I2C1SCL	CCP1	-	PWM0	PWM4	CAN0Rx	-	USB0EPEN	U1CTS	-	-
PA7	35	-	I2C1SDA	CCP4	-	PWM1	PWM5	CAN0Tx	CCP3	USB0PFLT	U1DCD	-	-
PB0	66	USB0ID	CCP0	PWM2	-	-	U1Rx	-	-	-	-	-	-
PB1	67	USB0VBUS	CCP2	PWM3	-	CCP1	U1Tx	-	-	-	-	-	-
PB2	72	-	I2C0SCL	IDX0	-	CCP3	CCP0	-	-	USB0EPEN	-	-	-
PB3	65	-	I2C0SDA	Fault0	-	Fault3	-	-	-	USB0PFLT	-	-	-
PB4	92	AIN10 C0-	-	-	-	U2Rx	CAN0Rx	IDX0	U1Rx	-	-	-	-
PB5	91	AIN11 C1-	C0o	CCP5	CCP6	CCP0	CAN0Tx	CCP2	U1Tx	-	-	-	-
PB6	90	VREFAC0+	CCP1	CCP7	C0o	Fault1	IDX0	CCP5	-	-	I2S0TXSCK	-	-
PB7	89	-	-	-	-	NMI	-	-	-	-	-	-	-
PC0	80	-	-	-	TCK SWCLK	-	-	-	-	-	-	-	-
PC1	79	-	-	-	TMS SWDIO	-	-	-	-	-	-	-	-

Table 9-2. GPIO Pins and Alternate Functions (100LQFP) (continued)

IO	Pin	Analog Function	Digital Function (GPIO_PCTL PMCx Bit Field Encoding) ^a										
			1	2	3	4	5	6	7	8	9	10	11
PC2	78	-	-	-	TDI	-	-	-	-	-	-	-	-
PC3	77	-	-	-	TDO SWO	-	-	-	-	-	-	-	-
PC4	25	-	CCP5	PhA0	-	-	CCP2	CCP4	-	-	CCP1	-	-
PC5	24	C1+	CCP1	Clo	C0o	Fault2	CCP3	USB0EPEN	-	-	-	-	-
PC6	23	-	CCP3	PhB0	-	-	U1Rx	CCP0	USB0PFLT	-	-	-	-
PC7	22	-	CCP4	PhB0	-	CCP0	U1Tx	USB0PFLT	Clo	-	-	-	-
PD0	10	AIN15	PWM0	CAN0RX	IDX0	U2Rx	U1Rx	CCP6	-	I2S0RXSCK	U1CTS	-	-
PD1	11	AIN14	PWM1	CAN0Tx	PhA0	U2Tx	U1Tx	CCP7	-	I2S0RXWS	U1DCD	CCP2	PhB1
PD2	12	AIN13	U1Rx	CCP6	PWM2	CCP5	-	-	-	-	-	-	-
PD3	13	AIN12	U1Tx	CCP7	PWM3	CCP0	-	-	-	-	-	-	-
PD4	97	AIN7	CCP0	CCP3	-	-	-	-	-	I2S0RXSD	U1RI	-	-
PD5	98	AIN6	CCP2	CCP4	-	-	-	-	-	I2S0RXM1K	U2Rx	-	-
PD6	99	AIN5	Fault0	-	-	-	-	-	-	I2S0IXSCK	U2Tx	-	-
PD7	100	AIN4	IDX0	C0o	CCP1	-	-	-	-	I2S0TXWS	U1DTR	-	-
PE0	74	-	PWM4	SSI1Clk	CCP3	-	-	-	-	-	USB0PFLT	-	-
PE1	75	-	PWM5	SSI1Fss	Fault0	CCP2	CCP6	-	-	-	-	-	-
PE2	95	AIN9	CCP4	SSI1Rx	PhB1	PhA0	CCP2	-	-	-	-	-	-
PE3	96	AIN8	CCP1	SSI1Tx	PhA1	PhB0	CCP7	-	-	-	-	-	-
PE4	6	AIN3	CCP3	-	-	Fault0	U2Tx	CCP2	-	-	I2S0TXWS	-	-
PE5	5	AIN2	CCP5	-	-	-	-	-	-	-	I2S0TXSD	-	-
PE6	2	AIN1	PWM4	Clo	-	-	-	-	-	-	U1CTS	-	-
PE7	1	AIN0	PWM5	-	-	-	-	-	-	-	U1DCD	-	-
PF0	47	-	CAN1RX	PhB0	PWM0	-	-	-	-	I2S0TXSD	U1DSR	-	-
PF1	61	-	CAN1Tx	IDX1	PWM1	-	-	-	-	I2S0IXM1K	U1RTS	CCP3	-
PF2	60	-	LED1	PWM4	-	PWM2	-	-	-	-	SSI1Clk	-	-
PF3	59	-	LED0	PWM5	-	PWM3	-	-	-	-	SSI1Fss	-	-
PF4	42	-	CCP0	C0o	-	Fault0	-	-	-	-	SSI1Rx	-	-
PF5	41	-	CCP2	Clo	-	-	-	-	-	-	SSI1Tx	-	-
PG0	19	-	U2Rx	PWM0	I2C1SCL	PWM4	-	-	USBOEPEN	-	-	-	-
PG1	18	-	U2Tx	PWM1	I2C1SDA	PWM5	-	-	-	-	-	-	-
PG7	36	-	PhB1	-	-	-	-	-	-	CCP5	-	-	-
PH0	86	-	CCP6	PWM2	-	-	-	-	-	-	PWM4	-	-
PH1	85	-	CCP7	PWM3	-	-	-	-	-	-	PWM5	-	-
PH2	84	-	IDX1	Clo	-	Fault3	-	-	-	-	-	-	-
PH3	83	-	PhB0	Fault0	-	USBOEPEN	-	-	-	-	-	-	-
PH4	76	-	-	-	-	USB0PFLT	-	-	-	-	-	-	SSI1Clk
PH5	63	-	-	-	-	-	-	-	-	-	Fault2	SSI1Fss	-
PH6	62	-	-	-	-	-	-	-	-	-	PWM4	SSI1Rx	-
PH7	15	-	-	-	-	-	-	-	-	-	PWM5	SSI1Tx	-
PJ0	14	-	-	-	-	-	-	-	-	-	PWM0	I2C1SCL	-

Table 9-2. GPIO Pins and Alternate Functions (100LQFP) (continued)

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding) ^a										
			1	2	3	4	5	6	7	8	9	10	11
PJ1	87	-	-	-	-	-	-	-	-	-	USB0PFLT	PWM1	I2C1SDA
PJ2	39	-	-	-	-	-	-	-	-	-	CCP0	Fault0	-

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin.

Table 9-3. GPIO Pins and Alternate Functions (108BGA)

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding) ^a										
			1	2	3	4	5	6	7	8	9	10	11
PA0	L3	-	U0Rx	-	-	-	-	-	-	I2C1SCL	U1Rx	-	-
PA1	M3	-	U0Tx	-	-	-	-	-	-	I2C1SDA	U1Tx	-	-
PA2	M4	-	SSI0Clk	-	-	PWM4	-	-	-	-	I2S0RXSD	-	-
PA3	L4	-	SSI0Fss	-	-	PWM5	-	-	-	-	I2S0RAMCLK	-	-
PA4	L5	-	SSI0RX	-	-	-	CAN0Rx	-	-	-	I2S0TXSCK	-	-
PA5	M5	-	SSI0TX	-	-	-	CAN0Tx	-	-	-	I2S0TXWS	-	-
PA6	L6	-	I2C1SCL	CCP1	-	PWM0	PWM4	CAN0Rx	-	USB0EPEN	U1CTS	-	-
PA7	M6	-	I2C1SDA	CCP4	-	PWM1	PWM5	CAN0Tx	CCP3	USB0PFLT	U1DCD	-	-
PB0	E12	USB0ID	CCP0	PWM2	-	-	U1Rx	-	-	-	-	-	-
PB1	D12	USB0VBUS	CCP2	PWM3	-	CCP1	U1Tx	-	-	-	-	-	-
PB2	A11	-	I2C0SCL	IDX0	-	CCP3	CCP0	-	-	USB0EPEN	-	-	-
PB3	E11	-	I2C0SDA	Fault0	-	Fault3	-	-	-	USB0PFLT	-	-	-
PB4	A6	AIN10 C0-	-	-	-	U2Rx	CAN0Rx	IDX0	U1Rx	-	-	-	-
PB5	B7	AIN11 C1-	C0o	CCP5	CCP6	CCP0	CAN0Tx	CCP2	U1Tx	-	-	-	-
PB6	A7	VREFA C0+	CCP1	CCP7	C0o	Fault1	IDX0	CCP5	-	-	I2S0TXSCK	-	-
PB7	A8	-	-	-	-	NMI	-	-	-	-	-	-	-
PC0	A9	-	-	-	TCK SWCLK	-	-	-	-	-	-	-	-
PC1	B9	-	-	-	TMS SWDIO	-	-	-	-	-	-	-	-
PC2	B8	-	-	-	TDI	-	-	-	-	-	-	-	-
PC3	A10	-	-	-	TDO_SWO	-	-	-	-	-	-	-	-
PC4	L1	-	CCP5	PhA0	-	-	CCP2	CCP4	-	-	CCP1	-	-
PC5	M1	C1+	CCP1	C1o	C0o	Fault2	CCP3	USB0EPEN	-	-	-	-	-
PC6	M2	-	CCP3	PhB0	-	-	U1Rx	CCP0	USB0PFLT	-	-	-	-
PC7	L2	-	CCP4	PhB0	-	CCP0	U1Tx	USB0PFLT	C1o	-	-	-	-
PD0	G1	AIN15	PWM0	CAN0Rx	IDX0	U2Rx	U1Rx	CCP6	-	I2S0RXSCK	U1CTS	-	-
PD1	G2	AIN14	PWM1	CAN0Tx	PhA0	U2Tx	U1Tx	CCP7	-	I2S0RXWS	U1DCD	CCP2	PhB1
PD2	H2	AIN13	U1Rx	CCP6	PWM2	CCP5	-	-	-	-	-	-	-
PD3	H1	AIN12	U1Tx	CCP7	PWM3	CCP0	-	-	-	-	-	-	-
PD4	B5	AIN7	CCP0	CCP3	-	-	-	-	-	I2S0RXSD	U1RI	-	-
PD5	C6	AIN6	CCP2	CCP4	-	-	-	-	-	I2S0RAMCLK	U2Rx	-	-

Table 9-3. GPIO Pins and Alternate Functions (108BGA) (continued)

IO	Pin	Analog Function	Digital Function (GPIO PCTL PMCx Bit Field Encoding) ^a										
			1	2	3	4	5	6	7	8	9	10	11
PD6	A3	AIN5	Fault0	-	-	-	-	-	-	I2S0TXSCK	U2Tx	-	-
PD7	A2	AIN4	IDX0	C0o	CCP1	-	-	-	-	I2S0TXWS	U1DTR	-	-
PE0	B11	-	PWM4	SSI1Clk	CCP3	-	-	-	-	-	USB0PFLT	-	-
PE1	A12	-	PWM5	SSI1Fss	Fault0	CCP2	CCP6	-	-	-	-	-	-
PE2	A4	AIN9	CCP4	SSI1Rx	PhB1	PhA0	CCP2	-	-	-	-	-	-
PE3	B4	AIN8	CCP1	SSI1Tx	PhA1	PhB0	CCP7	-	-	-	-	-	-
PE4	B2	AIN3	CCP3	-	-	Fault0	U2Tx	CCP2	-	-	I2S0TXWS	-	-
PE5	B3	AIN2	CCP5	-	-	-	-	-	-	-	I2S0TXSD	-	-
PE6	A1	AIN1	PWM4	C1o	-	-	-	-	-	-	U1CTS	-	-
PE7	B1	AIN0	PWM5	-	-	-	-	-	-	-	U1DCD	-	-
PF0	M9	-	CAN1Rx	PhB0	PWM0	-	-	-	-	I2S0TXSD	U1DSR	-	-
PF1	H12	-	CAN1Tx	IDX1	PWM1	-	-	-	-	I2S0TXM1K	U1RTS	CCP3	-
PF2	J11	-	LED1	PWM4	-	PWM2	-	-	-	-	SSI1Clk	-	-
PF3	J12	-	LED0	PWM5	-	PWM3	-	-	-	-	SSI1Fss	-	-
PF4	K4	-	CCP0	C0o	-	Fault0	-	-	-	-	SSI1Rx	-	-
PF5	K3	-	CCP2	C1o	-	-	-	-	-	-	SSI1Tx	-	-
PG0	K1	-	U2Rx	PWM0	I2C1SCL	PWM4	-	-	USBOEPEN	-	-	-	-
PG1	K2	-	U2Tx	PWM1	I2C1SDA	PWM5	-	-	-	-	-	-	-
PG7	C10	-	PhB1	-	-	-	-	-	-	CCP5	-	-	-
PH0	C9	-	CCP6	PWM2	-	-	-	-	-	-	PWM4	-	-
PH1	C8	-	CCP7	PWM3	-	-	-	-	-	-	PWM5	-	-
PH2	D11	-	IDX1	C1o	-	Fault3	-	-	-	-	-	-	-
PH3	D10	-	PhB0	Fault0	-	USBOEPEN	-	-	-	-	-	-	-
PH4	B10	-	-	-	-	USB0PFLT	-	-	-	-	-	-	SSI1Clk
PH5	F10	-	-	-	-	-	-	-	-	-	-	Fault2	SSI1Fss
PH6	G3	-	-	-	-	-	-	-	-	-	-	PWM4	SSI1Rx
PH7	H3	-	-	-	-	-	-	-	-	-	-	PWM5	SSI1Tx
PJ0	F3	-	-	-	-	-	-	-	-	-	-	PWM0	I2C1SCL
PJ1	B6	-	-	-	-	-	-	-	-	-	USB0PFLT	PWM1	I2C1SDA
PJ2	K6	-	-	-	-	-	-	-	-	-	-	CCP0	Fault0

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin.

9.2 Functional Description

Each GPIO port is a separate hardware instantiation of the same physical block (see Figure 9-1 on page 428 and Figure 9-2 on page 429). The LM3S9L97 microcontroller contains nine ports and thus nine of these physical GPIO blocks. Note that not all pins may be implemented on every block. 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 1190.

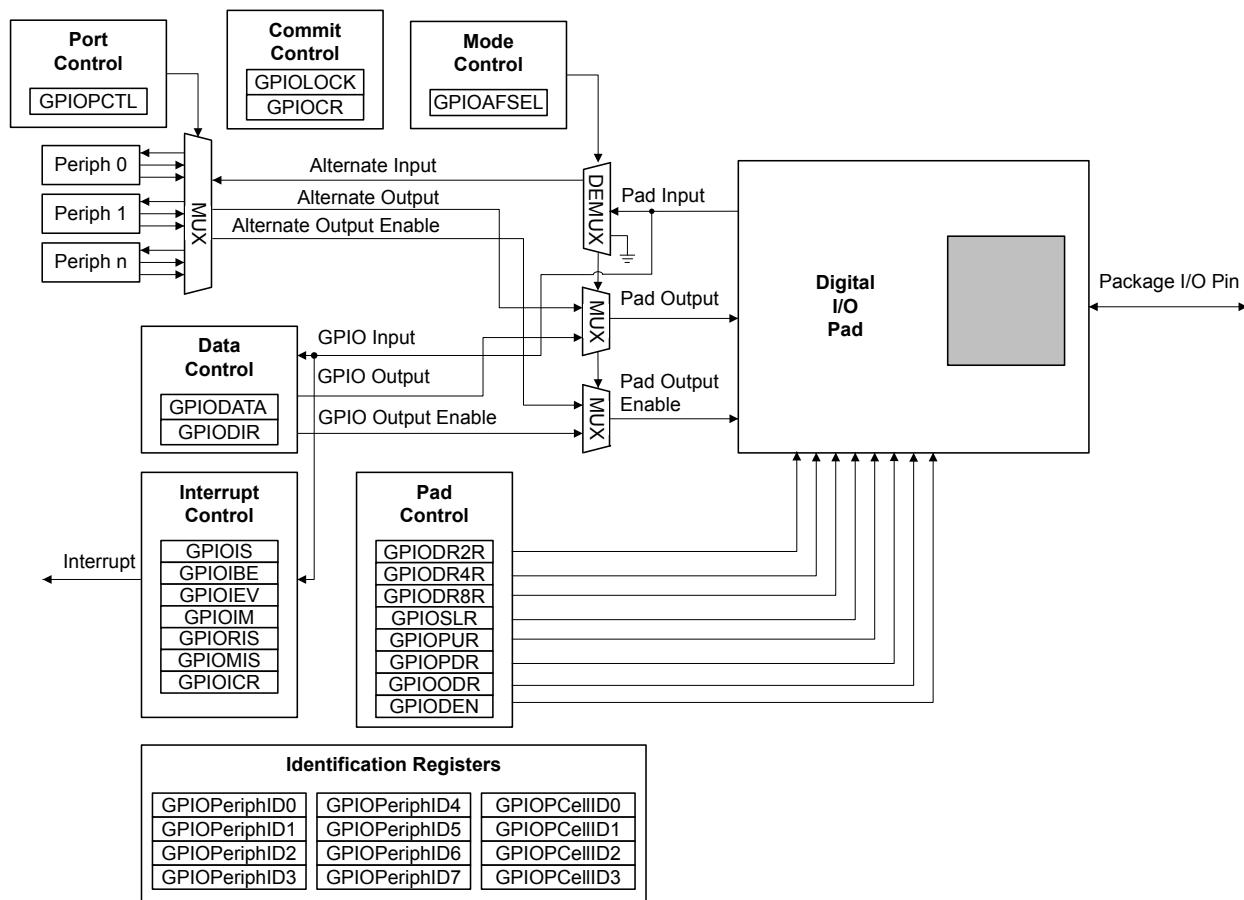
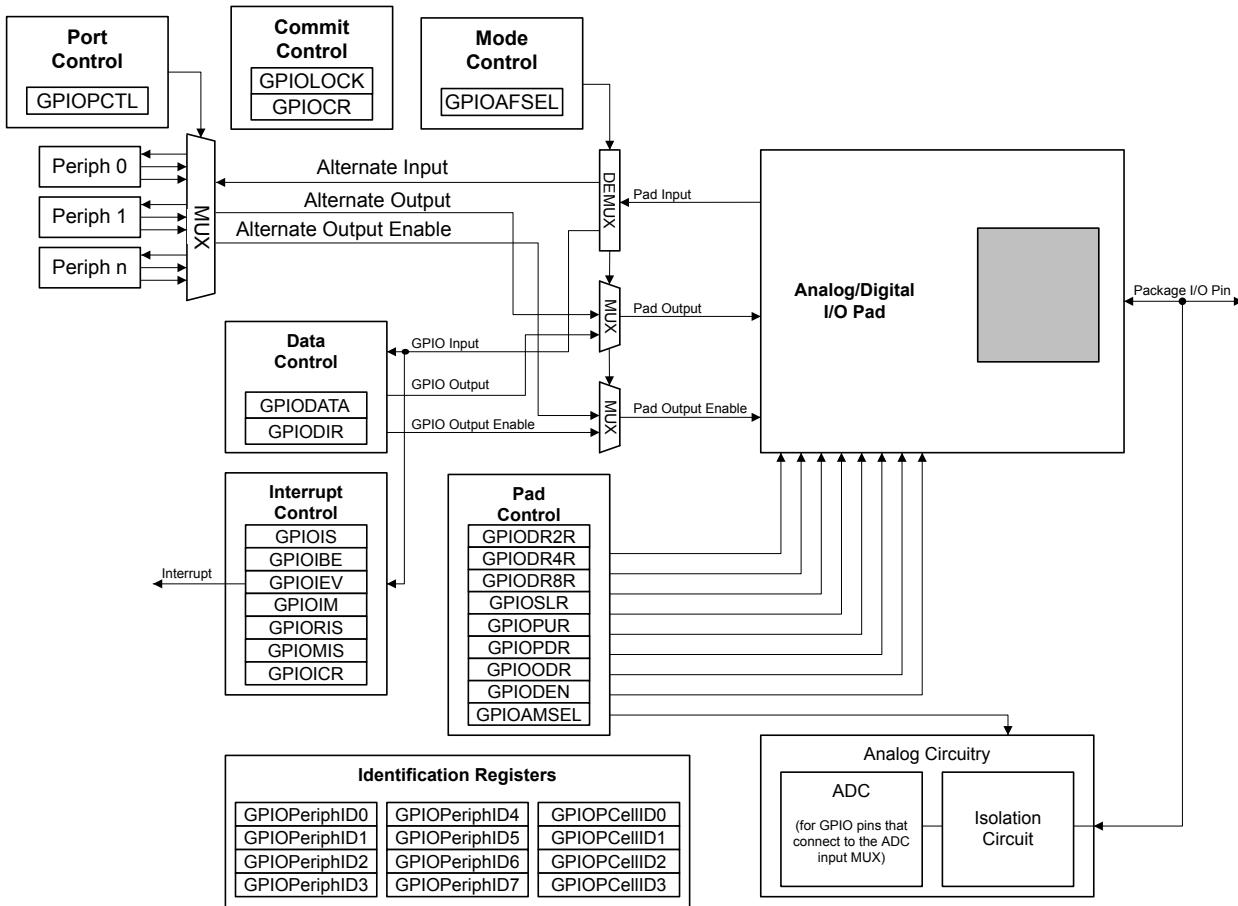
Figure 9-1. Digital I/O Pads

Figure 9-2. Analog/Digital I/O Pads

9.2.1 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.

9.2.1.1 Data Direction Operation

The **GPIO Direction (GPIODIR)** register (see page 438) 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.

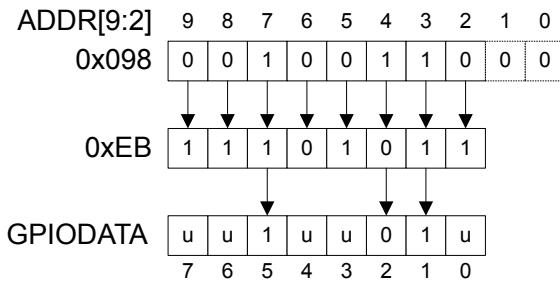
9.2.1.2 Data Register Operation

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 437) 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.

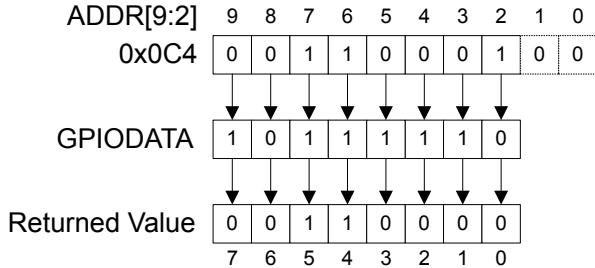
For example, writing a value of 0xEB to the address GPIODATA + 0x098 has the results shown in Figure 9-3, where *u* indicates that data is unchanged by the write.

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



9.2.2 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 439)

- **GPIO Interrupt Both Edges (GPIOIBE)** register (see page 440)
- **GPIO Interrupt Event (GPIOIEV)** register (see page 441)

Interrupts are enabled/disabled via the **GPIO Interrupt Mask (GPIOIM)** register (see page 442).

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 (GPIOIMIS)** registers (see page 443 and page 444). As the name implies, the **GPIOIMIS** 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.

Interrupts are cleared by writing a 1 to the appropriate bit of the **GPIO Interrupt Clear (GPIOICR)** register (see page 446).

When programming the interrupt control registers (**GPIOIS**, **GPIOIBE**, or **GPIOEV**), 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.

9.2.2.1 ADC Trigger Source

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 581.

If no other Port B pins are being used to generate interrupts, the **Interrupt 0-31 Set Enable (EN0)** 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 **EN0** register and the Port B interrupt handler must poll the ADC registers until the conversion is completed. See page 131 for more information.

9.2.3 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 **GPIODATA** 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 447), the pin state is controlled by its alternate function (that is, the peripheral).

Further pin muxing options are provided through the **GPIO Port Control (GPIOPCTL)** register which selects one of several peripheral functions for each GPIO. For information on the configuration options, refer to Table 24-5 on page 1190.

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.

9.2.4 Commit Control

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is provided for the NMI pin (PB7) and the four JTAG/SWD pins (PC[3:0]). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 447), **GPIO Pull Up Select (GPIOPUR)** register (see page 453), **GPIO Pull-Down Select (GPIOPDR)** register (see page 455), and **GPIO Digital Enable (GPIODEN)** register (see

page 458) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 460) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 461) have been set.

9.2.5 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**, **GPIOODR**, **GPIOPUR**, **GPIOPDR**, **GPIOSLR**, and **GPIODEN** registers. These registers control drive strength, open-drain configuration, pull-up and pull-down resistors, slew-rate control and digital 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 or BGA pin group with the total number of high-current GPIO outputs not exceeding four for the entire package.

9.2.6 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 **GPIOCellIID0-GPIOCellIID3** registers.

9.3 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 High-Performance 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 **GPIOHBCTL** register (see page 232).

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 285).

On reset, all GPIO pins are configured to be undriven (tristate): **GPIOAFSEL=0**, **GPIODEN=0**, **GPIOPDR=0**, and **GPIOPUR=0**, except for the pins shown in Table 9-1 on page 424. Table 9-4 on page 432 shows all possible configurations of the GPIO pads and the control register settings required to achieve them. Table 9-5 on page 433 shows how a rising edge interrupt is configured for pin 2 of a GPIO port.

Table 9-4. GPIO Pad Configuration Examples

Configuration	GPIO Register Bit Value ^a									
	AFSEL	DIR	ODR	DEN	PUR	PDR	DR2R	DR4R	DR8R	SLR
Digital Input (GPIO)	0	0	0	1	?	?	X	X	X	X
Digital Output (GPIO)	0	1	0	1	?	?	?	?	?	?
Open Drain Output (GPIO)	0	1	1	1	X	X	?	?	?	?
Open Drain Input/Output (I ² C)	1	X	1	1	X	X	?	?	?	?
Digital Input (Timer CCP)	1	X	0	1	?	?	X	X	X	X

Table 9-4. GPIO Pad Configuration Examples (continued)

Configuration	GPIO Register Bit Value ^a									
	AFSEL	DIR	ODR	DEN	PUR	PDR	DR2R	DR4R	DR8R	SLR
Digital Input (QEI)	1	X	0	1	?	?	X	X	X	X
Digital Output (PWM)	1	X	0	1	?	?	?	?	?	?
Digital Output (Timer PWM)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (SSI)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (UART)	1	X	0	1	?	?	?	?	?	?
Analog Input (Comparator)	0	0	0	0	0	0	X	X	X	X
Digital Output (Comparator)	1	X	0	1	?	?	?	?	?	?

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

?=Can be either 0 or 1, depending on the configuration

Table 9-5. GPIO Interrupt Configuration Example

Register	Desired Interrupt Event Trigger	Pin 2 Bit Value ^a								
		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)

9.4 Register Map

Table 9-7 on page 434 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 parts. The other aperture, the Advanced High-Performance 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 285).

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL=0**, **GPIODEN=0**, **GPIOPDR=0**, **GPIOPUR=0**, and **GPIOPCTL=0**, with the exception of the four JTAG/SWD pins (shown in the table below). A Power-On-Reset (**POR**) or asserting **RST** puts the pins back to their default state.

Table 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 ² C0	1	1	0	0	0x1
PC[3:0]	JTAG/SWD	1	1	0	1	0x3

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 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 GPIO pins, the **PC[3:0]** pins default to non-committable. Similarly, to ensure that the **NMI** pin is not accidentally programmed as a GPIO pin, the **PB7** pin 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 9-7. GPIO Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPIODATA	R/W	0x0000.0000	GPIO Data	437
0x400	GPIODIR	R/W	0x0000.0000	GPIO Direction	438
0x404	GPIOIS	R/W	0x0000.0000	GPIO Interrupt Sense	439

Table 9-7. GPIO Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x408	GPIOIBE	R/W	0x0000.0000	GPIO Interrupt Both Edges	440
0x40C	GPIOIEV	R/W	0x0000.0000	GPIO Interrupt Event	441
0x410	GPIOIM	R/W	0x0000.0000	GPIO Interrupt Mask	442
0x414	GPIORIS	RO	0x0000.0000	GPIO Raw Interrupt Status	443
0x418	GPIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status	444
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear	446
0x420	GPIOAFSEL	R/W	-	GPIO Alternate Function Select	447
0x500	GPIODR2R	R/W	0x0000.00FF	GPIO 2-mA Drive Select	449
0x504	GPIODR4R	R/W	0x0000.0000	GPIO 4-mA Drive Select	450
0x508	GPIODR8R	R/W	0x0000.0000	GPIO 8-mA Drive Select	451
0x50C	GPIOODR	R/W	0x0000.0000	GPIO Open Drain Select	452
0x510	GPIOPUR	R/W	-	GPIO Pull-Up Select	453
0x514	GPIOPDR	R/W	0x0000.0000	GPIO Pull-Down Select	455
0x518	GPIOSLR	R/W	0x0000.0000	GPIO Slew Rate Control Select	457
0x51C	GPIODEN	R/W	-	GPIO Digital Enable	458
0x520	GPIOLOCK	R/W	0x0000.0001	GPIO Lock	460
0x524	GPIOCR	-	-	GPIO Commit	461
0x528	GPIOAMSEL	R/W	0x0000.0000	GPIO Analog Mode Select	463
0x52C	GPIOPCTL	R/W	-	GPIO Port Control	465
0xFD0	GPIOPeriphID4	RO	0x0000.0000	GPIO Peripheral Identification 4	467
0xFD4	GPIOPeriphID5	RO	0x0000.0000	GPIO Peripheral Identification 5	468
0xFD8	GPIOPeriphID6	RO	0x0000.0000	GPIO Peripheral Identification 6	469
0xFDC	GPIOPeriphID7	RO	0x0000.0000	GPIO Peripheral Identification 7	470
0xFE0	GPIOPeriphID0	RO	0x0000.0061	GPIO Peripheral Identification 0	471
0xFE4	GPIOPeriphID1	RO	0x0000.0000	GPIO Peripheral Identification 1	472
0xFE8	GPIOPeriphID2	RO	0x0000.0018	GPIO Peripheral Identification 2	473
0xFEC	GPIOPeriphID3	RO	0x0000.0001	GPIO Peripheral Identification 3	474
0xFF0	GPIOPCellID0	RO	0x0000.000D	GPIO PrimeCell Identification 0	475
0xFF4	GPIOPCellID1	RO	0x0000.00F0	GPIO PrimeCell Identification 1	476
0xFF8	GPIOPCellID2	RO	0x0000.0005	GPIO PrimeCell Identification 2	477
0xFFC	GPIOPCellID3	RO	0x0000.00B1	GPIO PrimeCell Identification 3	478

9.5 Register Descriptions

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

Register 1: GPIO Data (GPIO DATA), offset 0x000

The **GPIO DATA** register is the data register. In software control mode, values written in the **GPIO DATA** register are transferred onto the GPIO port pins if the respective pins have been configured as outputs through the **GPIO Direction (GPIO DIR)** register (see page 438).

In order to write to **GPIO DATA**, 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 **GPIO DATA** to be read, and bits that are clear in the address mask cause the corresponding bits in **GPIO DATA** to be read as 0, regardless of their value.

A read from **GPIO DATA** 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 (GPIO DATA)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	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 430 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

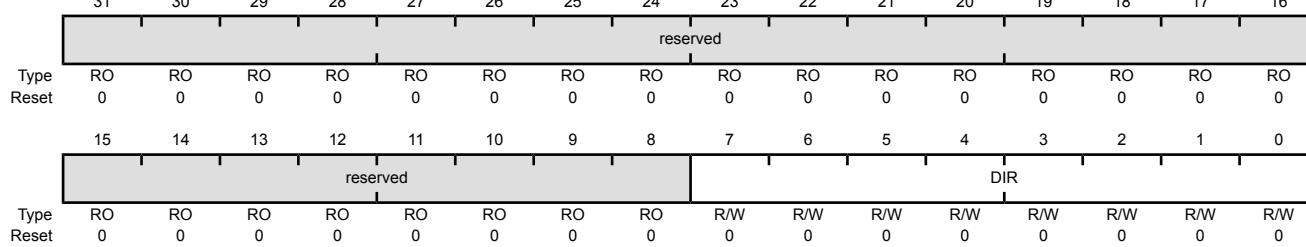
GPIO Port H (AHB) base: 0x4005.F000

GPIO Port J (APB) base: 0x4003.D000

GPIO Port J (AHB) base: 0x4006.0000

Offset 0x400

Type R/W, reset 0x0000.0000



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

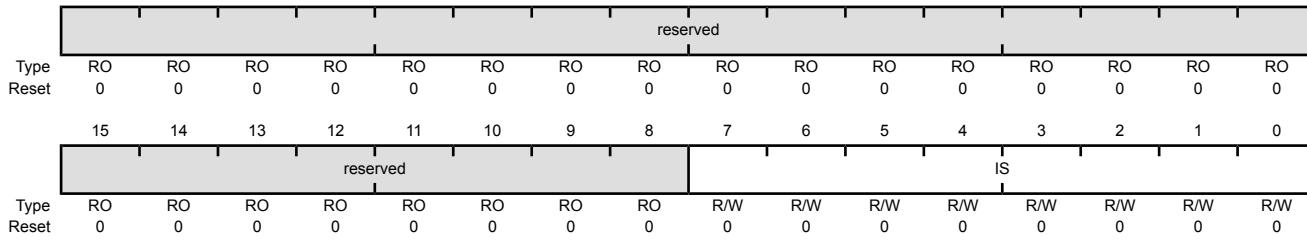
GPIO Port H (AHB) base: 0x4005.F000

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 439) 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 441). 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

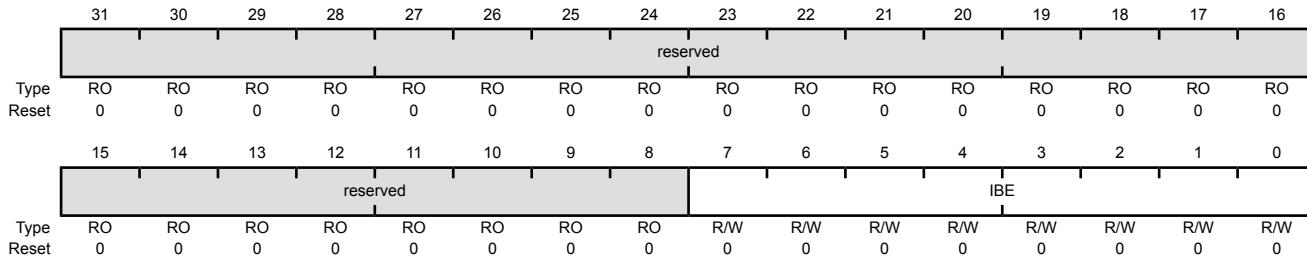
GPIO Port H (AHB) base: 0x4005.F000

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 GPIO Interrupt Event (GPIOIEV) register (see page 441).
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 439). 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

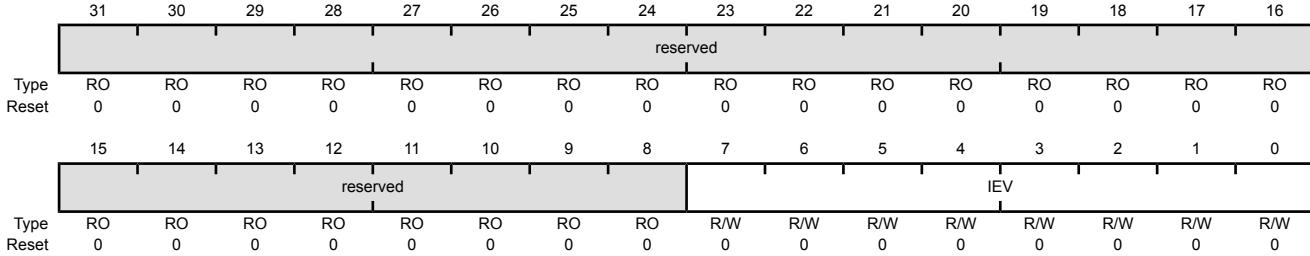
GPIO Port H (AHB) base: 0x4005.F000

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

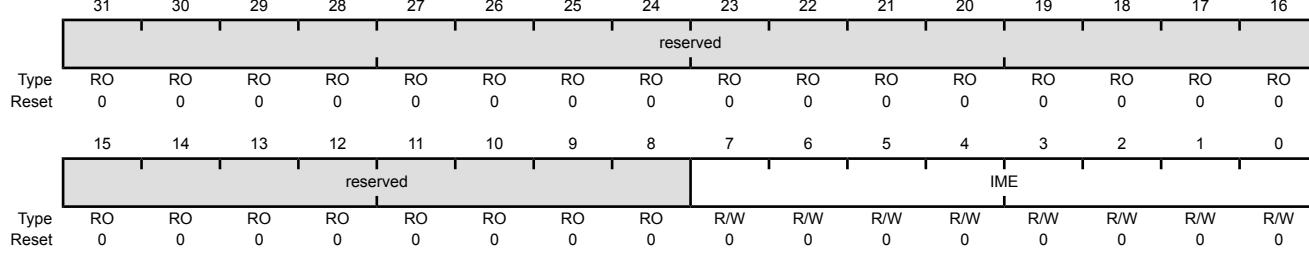
GPIO Port H (AHB) base: 0x4005.F000

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 (GPIOISR), offset 0x414

The **GPIOISR** 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 442) 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 (GPIOISR)

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

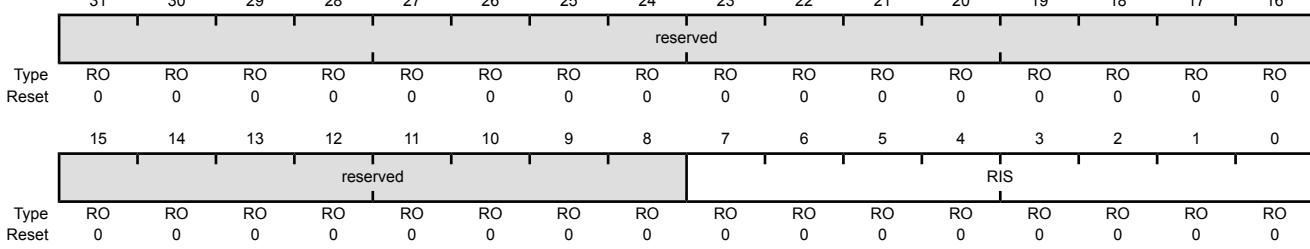
GPIO Port H (AHB) base: 0x4005.F000

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 (GPIO MIS), offset 0x418

The **GPIO MIS** 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 581.

If no other Port B pins are being used to generate interrupts, the **Interrupt 0-31 Set Enable (EN0)** 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 **EN0** register and the Port B interrupt handler must poll the ADC registers until the conversion is completed. See page 131 for more information.

GPIO MIS is the state of the interrupt after masking.

GPIO Masked Interrupt Status (GPIO MIS)

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

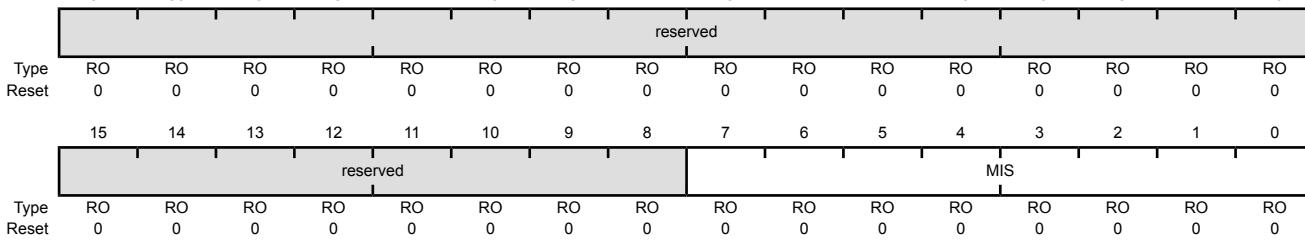
GPIO Port H (AHB) base: 0x4005.F000

GPIO Port J (APB) base: 0x4003.D000

GPIO Port J (AHB) base: 0x4006.0000

Offset 0x418

Type RO, reset 0x0000.0000



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						
<table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>1</td><td>An interrupt condition on the corresponding pin has triggered an interrupt to the interrupt controller.</td></tr><tr><td>0</td><td>An interrupt condition on the corresponding pin is masked or has not occurred.</td></tr></tbody></table>					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.
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 GPIOICR 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 **GPIORIS** and **GPIOVIS** 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

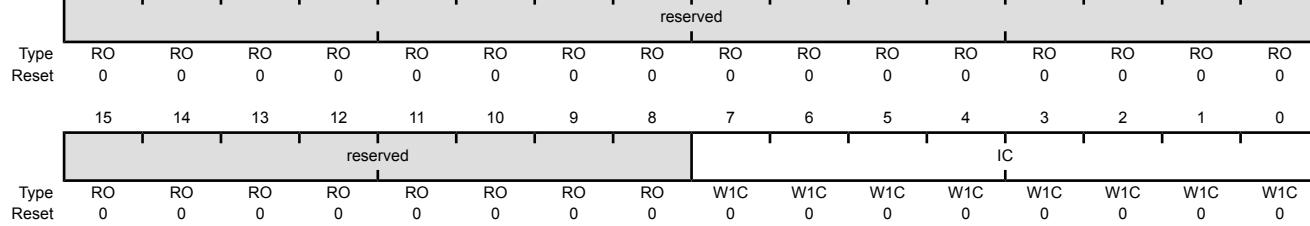
GPIO Port H (AHB) base: 0x4005.F000

GPIO Port J (APB) base: 0x4003.D000

GPIO Port J (AHB) base: 0x4006.0000

Offset 0x41C

Type W1C, reset 0x0000.0000



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 1190 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 the table below.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the four JTAG/SWD pins (shown in the table below). A Power-On-Reset (**POR**) or asserting **RST** puts the pins back to their default state.

Table 9-8. 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 ² C0	1	1	0	0	0x1
PC[3:0]	JTAG/SWD	1	1	0	1	0x3

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 provided for the **NMI** pin (**PB7**) and the four JTAG/SWD pins (**PC[3:0]**). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 447), **GPIO Pull Up Select (GPIOPUR)** register (see page 453), **GPIO Pull-Down Select (GPIOPDR)** register (see page 455), and **GPIO Digital Enable (GPIODEN)** register (see page 458) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 460) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 461) have been set.

When using the I²C module, in addition to setting the **GPIOAFSEL** register bits for the I²C 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 432).

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

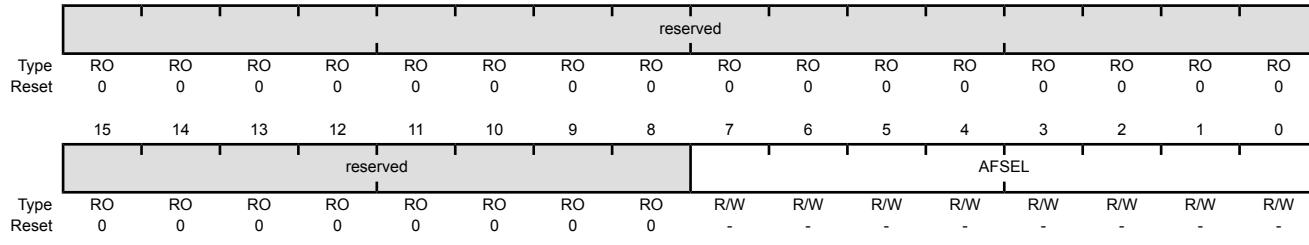
GPIO Port H (AHB) base: 0x4005.F000

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 424.

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

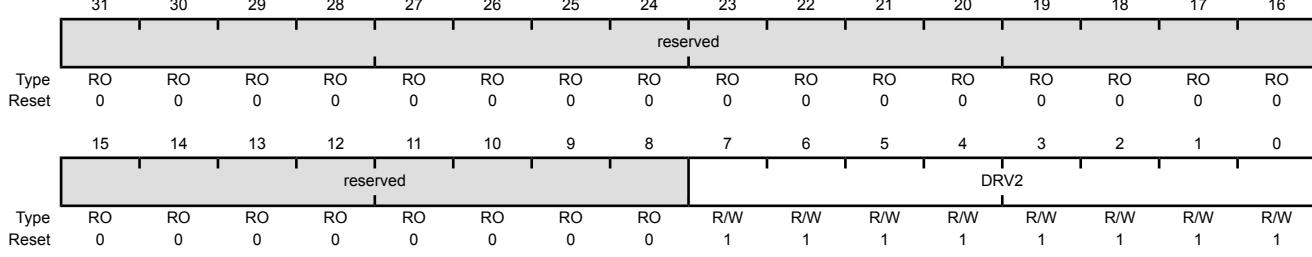
GPIO Port H (AHB) base: 0x4005.F000

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

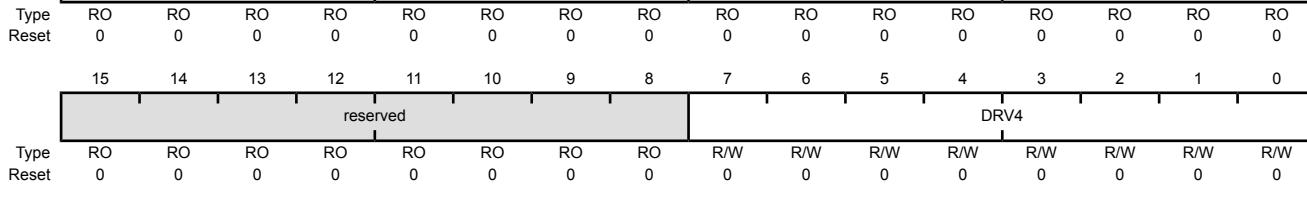
GPIO Port H (AHB) base: 0x4005.F000

GPIO Port J (APB) base: 0x4003.D000

GPIO Port J (AHB) base: 0x4006.0000

Offset 0x504

Type R/W, reset 0x0000.0000



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 1232 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

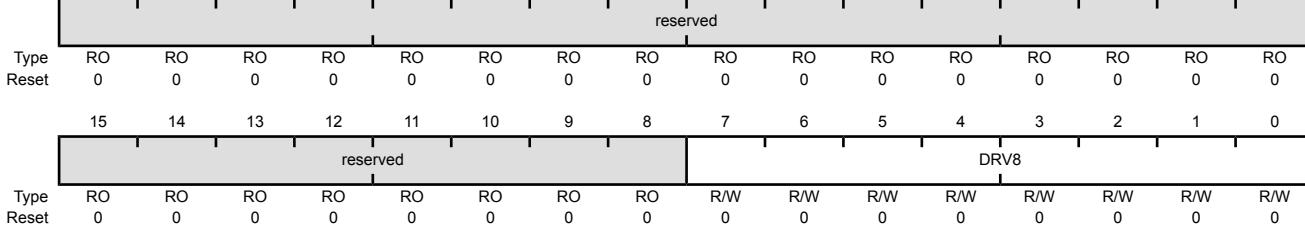
GPIO Port H (AHB) base: 0x4005.F000

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 (GPIOODR), offset 0x50C

The **GPIOODR** register is the open drain control register. Setting a bit in this register enables the open-drain configuration of the corresponding GPIO pad. When open-drain mode is enabled, the corresponding bit should also be set in the **GPIO Digital Input Enable (GPIODEN)** register (see page 458). 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. If open drain is selected while the GPIO is configured as an input, the GPIO will remain an input and the open-drain selection has no effect until the GPIO is changed to an output.

When using the I²C module, in addition to configuring the pin to open drain, the **GPIO Alternate Function Select (GPIOAFSEL)** register bits for the I²C clock and data pins should be set (see examples in “Initialization and Configuration” on page 432).

GPIO Open Drain Select (GPIOODR)

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

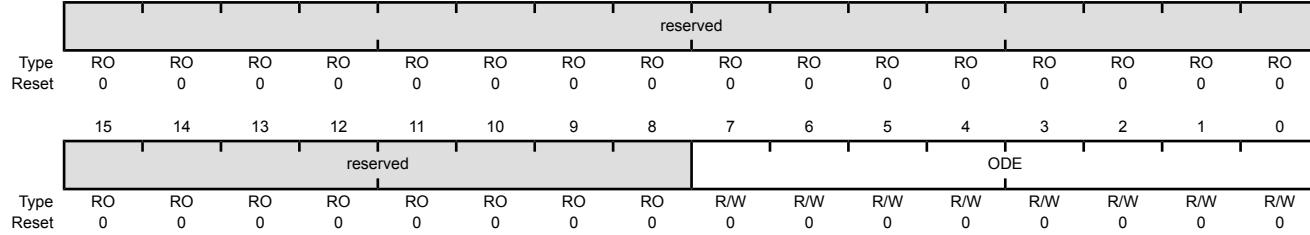
GPIO Port H (AHB) base: 0x4005.F000

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 455). 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, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the four JTAG/SWD pins (shown in the table below). A Power-On-Reset (**POR**) or asserting **RST** puts the pins back to their default state.

Table 9-9. 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 ² C0	1	1	0	0	0x1
PC[3:0]	JTAG/SWD	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 provided for the **NMI** pin (**PB7**) and the four JTAG/SWD pins (**PC[3:0]**). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 447), **GPIO Pull Up Select (GPIOPUR)** register (see page 453), **GPIO Pull-Down Select (GPIOPDR)** register (see page 455), and **GPIO Digital Enable (GPIODEN)** register (see page 458) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 460) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 461) 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

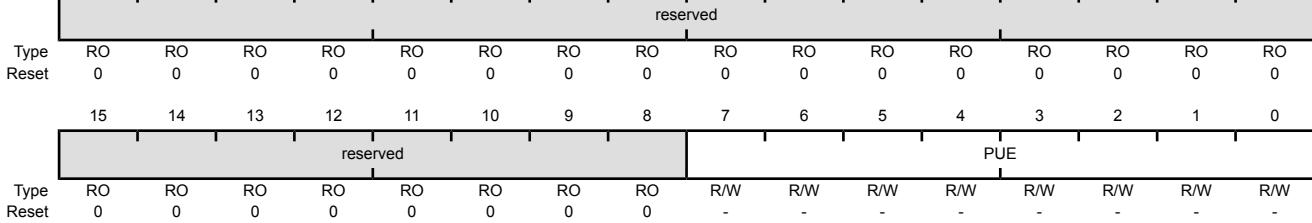
GPIO Port H (AHB) base: 0x4005.F000

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	-	Pad Weak Pull-Up Enable						
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The corresponding pin has a weak pull-up resistor.</td></tr> <tr> <td>0</td><td>The corresponding pin is not affected.</td></tr> </tbody> </table>	Value	Description	1	The corresponding pin has a weak pull-up resistor.	0	The corresponding pin is not affected.
Value	Description									
1	The corresponding pin has a weak pull-up resistor.									
0	The corresponding pin is not affected.									
				Setting a bit in the GPIOPDR register clears the corresponding bit in the GPIOPUR 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. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 9-1 on page 424.						

Register 16: GPIO Pull-Down Select (GPIOOPDR), offset 0x514

The **GPIOOPDR** 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 **GPIOOPDR** automatically clears the corresponding bit in the **GPIO Pull-Up Select (GPIOPUR)** register (see page 453).

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the four JTAG/SWD pins (shown in the table below). A Power-On-Reset (**POR**) or asserting **RST** puts the pins back to their default state.

Table 9-10. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	1	1	0	0	0x1
PA[5:2]	SSIO	1	1	0	0	0x1
PB[3:2]	I ² C0	1	1	0	0	0x1
PC[3:0]	JTAG/SWD	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 provided for the **NMI** pin (**PB7**) and the four JTAG/SWD pins (**PC[3:0]**). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 447), **GPIO Pull Up Select (GPIOPUR)** register (see page 453), **GPIO Pull-Down Select (GPIOOPDR)** register (see page 455), and **GPIO Digital Enable (GPIODEN)** register (see page 458) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 460) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 461) have been set.

GPIO Pull-Down Select (GPIOOPDR)

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

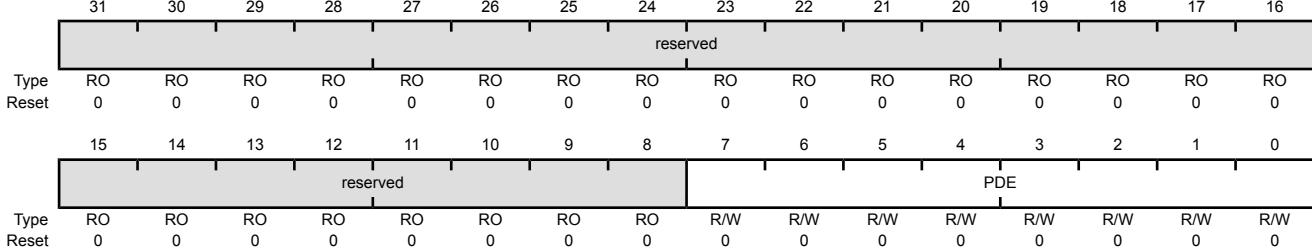
GPIO Port H (AHB) base: 0x4005.F000

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						
				<table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>1</td><td>The corresponding pin has a weak pull-down resistor.</td></tr><tr><td>0</td><td>The corresponding pin is not affected.</td></tr></tbody></table>	Value	Description	1	The corresponding pin has a weak pull-down resistor.	0	The corresponding pin is not affected.
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 451).

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							
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, all GPIO signals except those listed below 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 as a digital input or output (either GPIO or alternate function), the corresponding GPIODEN bit must be set.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the four JTAG/SWD pins (shown in the table below). A Power-On-Reset (**POR**) or asserting **RST** puts the pins back to their default state.

Table 9-11. 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]	SSIO	1	1	0	0	0x1
PB[3:2]	I ² C0	1	1	0	0	0x1
PC[3:0]	JTAG/SWD	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 provided for the **NMI** pin (**PB7**) and the four JTAG/SWD pins (**PC[3:0]**). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 447), **GPIO Pull Up Select (GPIOPUR)** register (see page 453), **GPIO Pull-Down Select (GPIOPDR)** register (see page 455), and **GPIO Digital Enable (GPIODEN)** register (see page 458) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 460) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 461) 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

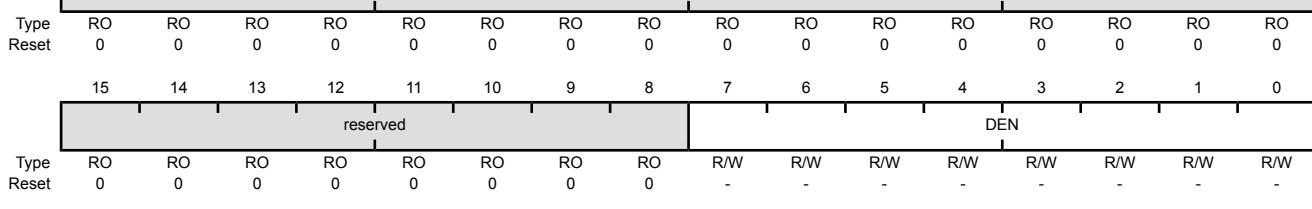
GPIO Port H (AHB) base: 0x4005.F000

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.

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 424.

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

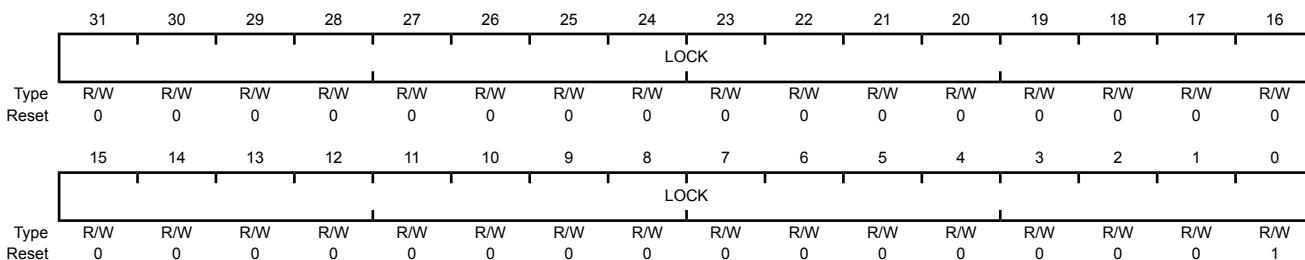
The **GPIOLOCK** register enables write access to the **GPIOCR** register (see page 461). Writing 0x4C4F.434B to the **GPIOLOCK** register unlocks the **GPIOCR** register. Writing any other value to the **GPIOLOCK** register re-enables the locked state. Reading the **GPIOLOCK** register returns the lock 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 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
0x1	The GPIOCR register is locked and may not be modified.
0x0	The GPIOCR 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 **GPIOLOCK** register is unlocked. Writes to the **GPIOCR** register are ignored if the status in the **GPIOLOCK** register is locked.

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

Because this protection is currently only implemented on the NMI and JTAG/SWD pins on PB7 and PC[3:0], all of the other bits in the **GPIOCR** registers cannot be written with 0x0. These bits are hardwired to 0x1, ensuring that it is always possible to commit new values to the **GPIOAFSEL**, **GPIOPUR**, **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															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR																
Type	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 GPIODEN bits can be written.
			0	The corresponding GPIOAFSEL , GPIOPUR , GPIOPDR , or GPIODEN 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 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 GPIO pins, the PC[3:0] pins default to non-committable. Similarly, to ensure that the NMI pin is not accidentally programmed as a GPIO pin, the PB7 pin 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 1190.

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

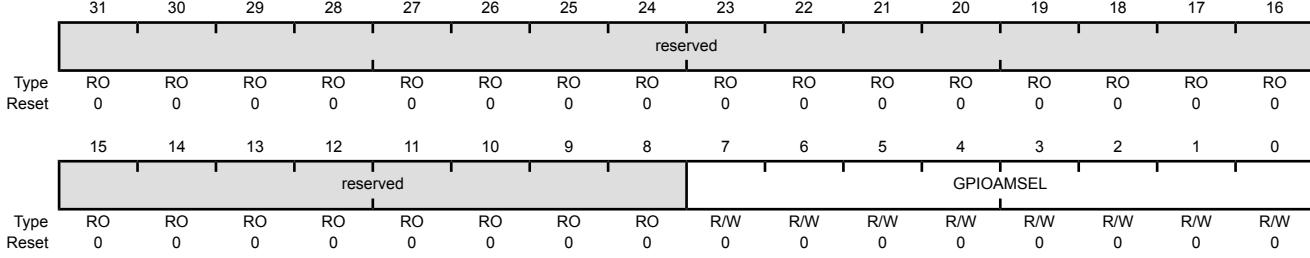
GPIO Port H (AHB) base: 0x4005.F000

GPIO Port J (APB) base: 0x4003.D000

GPIO Port J (AHB) base: 0x4006.0000

Offset 0x528

Type R/W, reset 0x0000.0000



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	GPIOAMSEL	R/W	0x00	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.	
Note: 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.				

Register 22: GPIO Port Control (GPIOPCTL), offset 0x52C

The **GPIOPCTL** 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 on 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 **GPIOPCTL** register selects one out of a set of peripheral functions for each GPIO, providing additional flexibility in signal definition. For information on the defined encodings for the bit fields in this register, refer to Table 24-5 on page 1190. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in the table below.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0, with the exception of the four JTAG/SWD pins (shown in the table below). A Power-On-Reset (**POR**) or asserting **RST** puts the pins back to their default state.

Table 9-12. 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 ² C0	1	1	0	0	0x1
PC[3:0]	JTAG/SWD	1	1	0	1	0x3

GPIO Port Control (GPIOPCTL)

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 -

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PMC7				PMC6				PMC5				PMC4			
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
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PMC3				PMC2				PMC1				PMC0			
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
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.

Bit/Field	Name	Type	Reset	Description
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

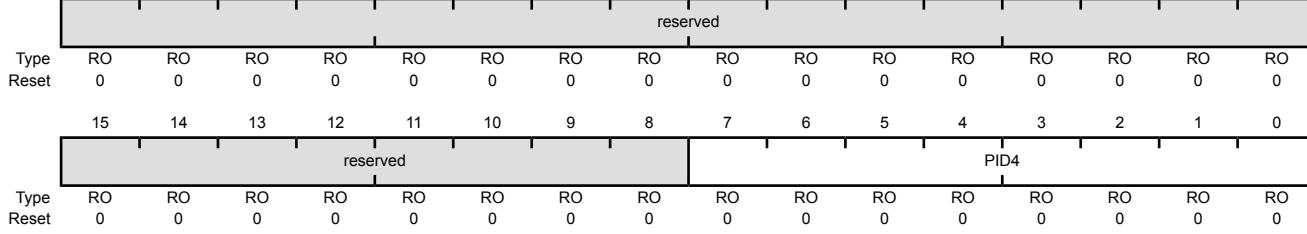
GPIO Port H (AHB) base: 0x4005.F000

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

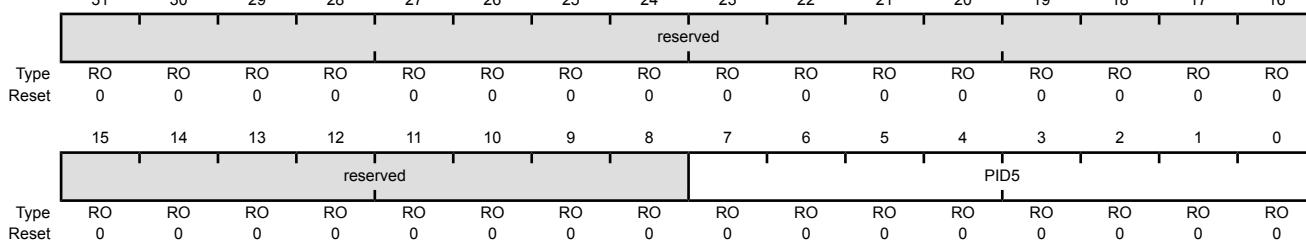
GPIO Port H (AHB) base: 0x4005.F000

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

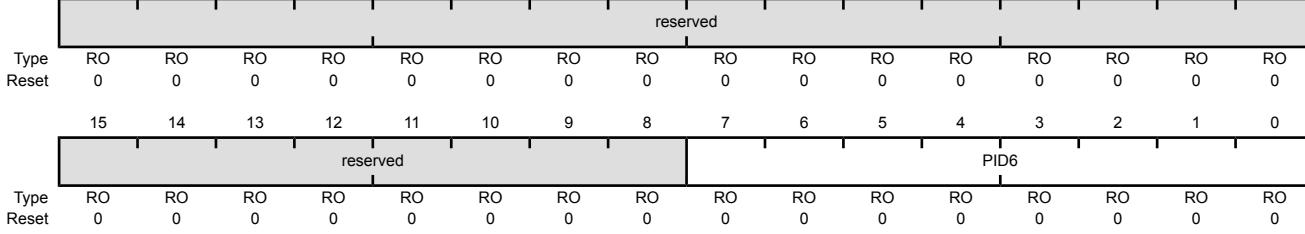
GPIO Port H (AHB) base: 0x4005.F000

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

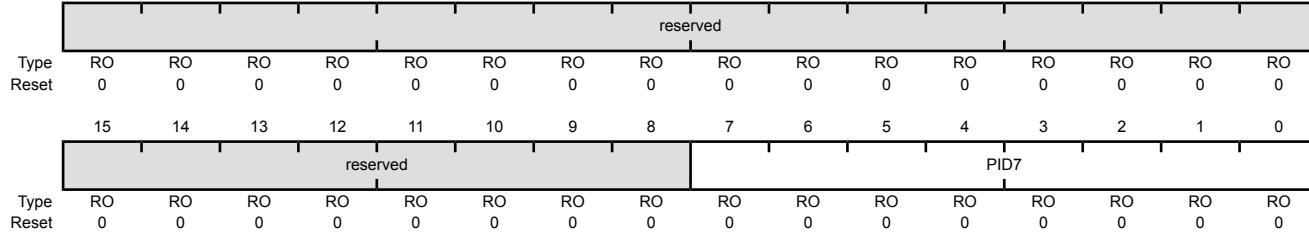
GPIO Port H (AHB) base: 0x4005.F000

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

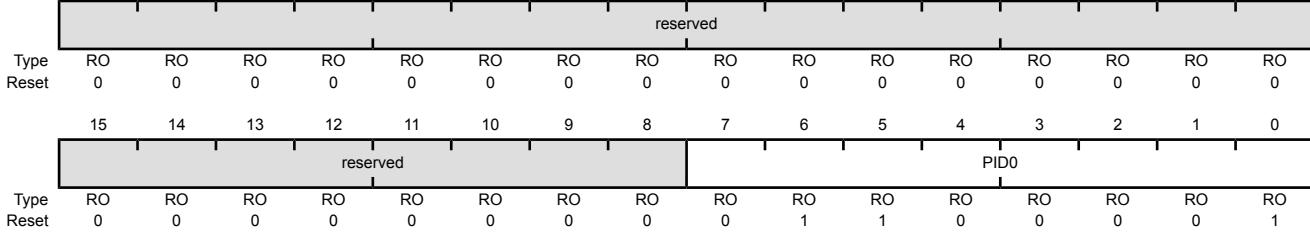
GPIO Port H (AHB) base: 0x4005.F000

GPIO Port J (APB) base: 0x4003.D000

GPIO Port J (AHB) base: 0x4006.0000

Offset 0xFE0

Type RO, reset 0x0000.0061



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

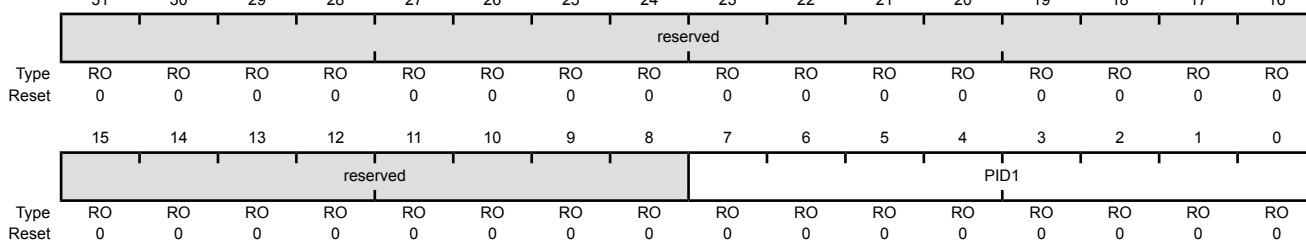
GPIO Port H (AHB) base: 0x4005.F000

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

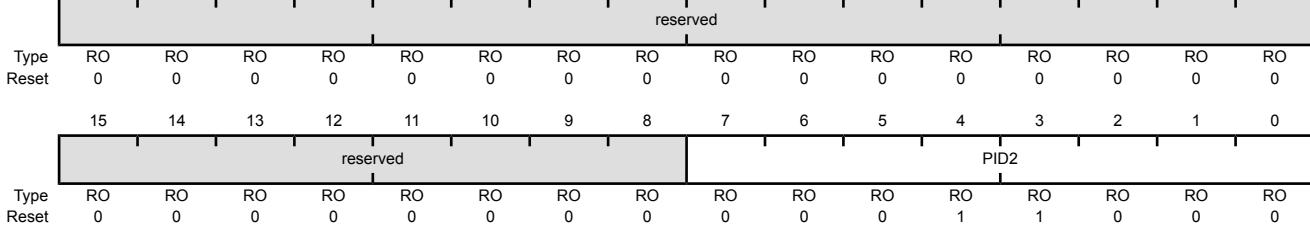
GPIO Port H (AHB) base: 0x4005.F000

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

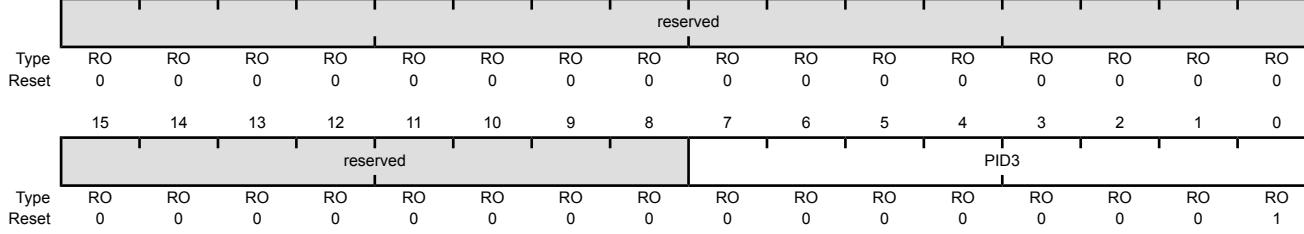
GPIO Port H (AHB) base: 0x4005.F000

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

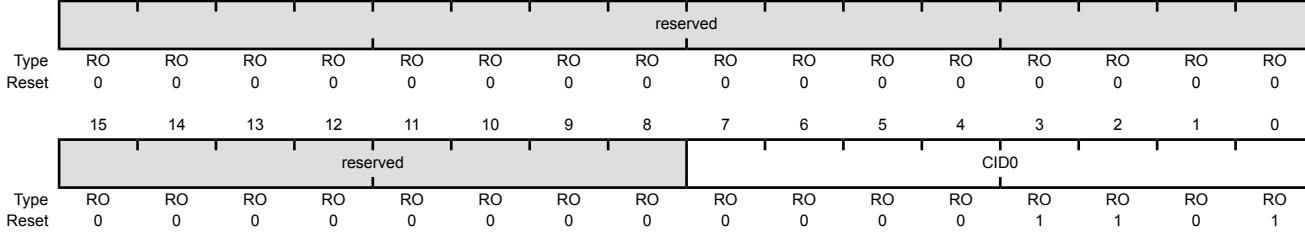
GPIO Port H (AHB) base: 0x4005.F000

GPIO Port J (APB) base: 0x4003.D000

GPIO Port J (AHB) base: 0x4006.0000

Offset 0xFF0

Type RO, reset 0x0000.000D



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

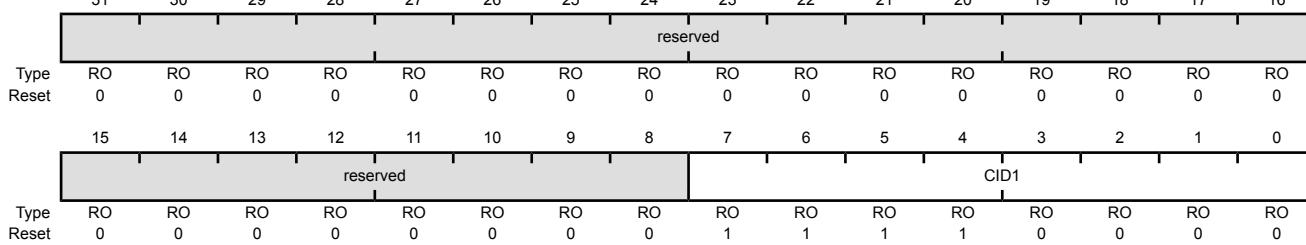
GPIO Port H (AHB) base: 0x4005.F000

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

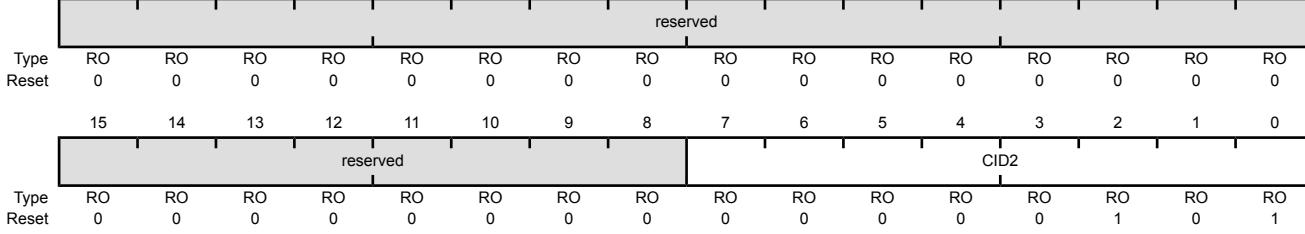
GPIO Port H (AHB) base: 0x4005.F000

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

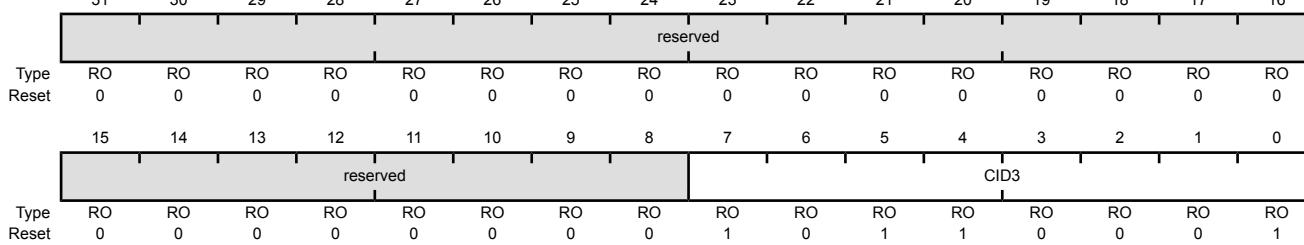
GPIO Port H (AHB) base: 0x4005.F000

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 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. 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 concatenated to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC). Timers can also be used to trigger µDMA 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 116) and the PWM timer in the PWM module (see “PWM Timer” on page 1070).

The General-Purpose Timer Module (GPTM) contains four GPTM blocks with the following functional options:

- Operating modes:
 - 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
 - 16-bit input-edge count- or time-capture modes
 - 16-bit PWM mode with software-programmable output inversion of the PWM signal
- Count up or down
- Eight Capture Compare PWM pins (CCP)
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events
- ADC event trigger
- User-enabled stalling when the microcontroller asserts CPU Halt flag during debug (excluding RTC mode)
- 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 (µDMA)
 - Dedicated channel for each timer
 - Burst request generated on timer interrupt

10.1 Block Diagram

In the block diagram, the specific Capture Compare PWM (CCP) pins available depend on the Stellaris device. See Table 10-1 on page 480 for the available CCP pins and their timer assignments.

Figure 10-1. GPTM Module Block Diagram

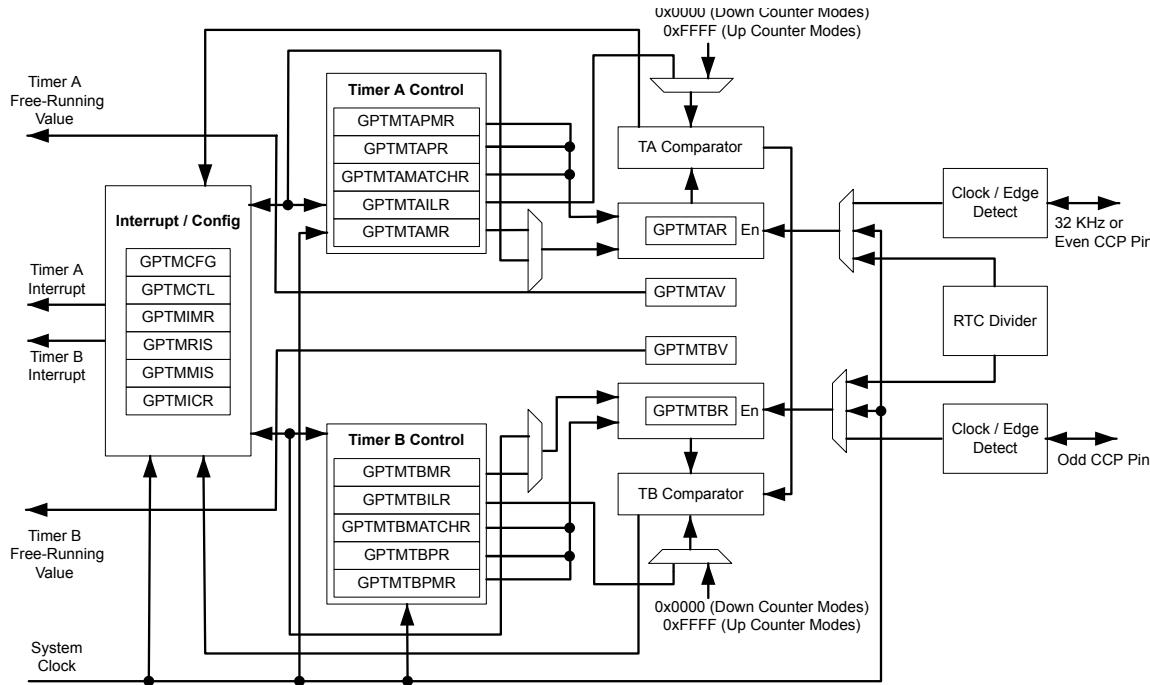


Table 10-1. Available CCP Pins

Timer	16-Bit Up/Down Counter	Even CCP Pin	Odd CCP Pin
Timer 0	TimerA	CCP0	-
	TimerB	-	CCP1
Timer 1	TimerA	CCP2	-
	TimerB	-	CCP3
Timer 2	TimerA	CCP4	-
	TimerB	-	CCP5
Timer 3	TimerA	CCP6	-
	TimerB	-	CCP7

10.2 Signal Description

Table 10-2 on page 481 and Table 10-3 on page 482 list the external signals of the GP Timer module and describe the function of each. The GP Timer signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these GP Timer signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 447) should be set to choose the GP Timer function. The number in parentheses is the encoding that must be programmed into

the **PMCn** field in the **GPIO Port Control (GPIOPCTL)** register (page 465) to assign the GP Timer signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 423.

Table 10-2. Signals for General-Purpose Timers (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
CCP0	13 22 23 39 42 66 72 91 97	PD3 (4) PC7 (4) PC6 (6) PJ2 (9) PF4 (1) PB0 (1) PB2 (5) PB5 (4) PD4 (1)	I/O	TTL	Capture/Compare/PWM 0.
CCP1	24 25 34 67 90 96 100	PC5 (1) PC4 (9) PA6 (2) PB1 (4) PB6 (1) PE3 (1) PD7 (3)	I/O	TTL	Capture/Compare/PWM 1.
CCP2	6 11 25 41 67 75 91 95 98	PE4 (6) PD1 (10) PC4 (5) PF5 (1) PB1 (1) PE1 (4) PB5 (6) PE2 (5) PD5 (1)	I/O	TTL	Capture/Compare/PWM 2.
CCP3	6 23 24 35 61 72 74 97	PE4 (1) PC6 (1) PC5 (5) PA7 (7) PF1 (10) PB2 (4) PE0 (3) PD4 (2)	I/O	TTL	Capture/Compare/PWM 3.
CCP4	22 25 35 95 98	PC7 (1) PC4 (6) PA7 (2) PE2 (1) PD5 (2)	I/O	TTL	Capture/Compare/PWM 4.
CCP5	5 12 25 36 90 91	PE5 (1) PD2 (4) PC4 (1) PG7 (8) PB6 (6) PB5 (2)	I/O	TTL	Capture/Compare/PWM 5.
CCP6	10 12 75 86 91	PD0 (6) PD2 (2) PE1 (5) PH0 (1) PB5 (3)	I/O	TTL	Capture/Compare/PWM 6.

Table 10-2. Signals for General-Purpose Timers (100LQFP) (continued)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
CCP7	11 13 85 90 96	PD1 (6) PD3 (2) PH1 (1) PB6 (2) PE3 (5)	I/O	TTL	Capture/Compare/PWM 7.

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

Table 10-3. Signals for General-Purpose Timers (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
CCP0	H1 L2 M2 K6 K4 E12 A11 B7 B5	PD3 (4) PC7 (4) PC6 (6) PJ2 (9) PF4 (1) PB0 (1) PB2 (5) PB5 (4) PD4 (1)	I/O	TTL	Capture/Compare/PWM 0.
CCP1	M1 L1 L6 D12 A7 B4 A2	PC5 (1) PC4 (9) PA6 (2) PB1 (4) PB6 (1) PE3 (1) PD7 (3)	I/O	TTL	Capture/Compare/PWM 1.
CCP2	B2 G2 L1 K3 D12 A12 B7 A4 C6	PE4 (6) PD1 (10) PC4 (5) PF5 (1) PB1 (1) PE1 (4) PB5 (6) PE2 (5) PD5 (1)	I/O	TTL	Capture/Compare/PWM 2.
CCP3	B2 M2 M1 M6 H12 A11 B11 B5	PE4 (1) PC6 (1) PC5 (5) PA7 (7) PF1 (10) PB2 (4) PE0 (3) PD4 (2)	I/O	TTL	Capture/Compare/PWM 3.
CCP4	L2 L1 M6 A4 C6	PC7 (1) PC4 (6) PA7 (2) PE2 (1) PD5 (2)	I/O	TTL	Capture/Compare/PWM 4.
CCP5	B3 H2 L1 C10 A7 B7	PE5 (1) PD2 (4) PC4 (1) PG7 (8) PB6 (6) PB5 (2)	I/O	TTL	Capture/Compare/PWM 5.

Table 10-3. Signals for General-Purpose Timers (108BGA) (continued)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
CCP6	G1 H2 A12 C9 B7	PD0 (6) PD2 (2) PE1 (5) PH0 (1) PB5 (3)	I/O	TTL	Capture/Compare/PWM 6.
CCP7	G2 H1 C8 A7 B4	PD1 (6) PD3 (2) PH1 (1) PB6 (2) PE3 (5)	I/O	TTL	Capture/Compare/PWM 7.

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

10.3 Functional Description

The main components of each GPTM block are two free-running up/down counters (referred to as Timer A and Timer B), two match registers, two prescaler match registers, two shadow registers, and two load/initialization registers and their associated control functions. The exact functionality of each GPTM is controlled by software and configured through the register interface. Timer A and Timer B can be used individually, in which case they have a 16-bit counting range. In addition, Timer A and Timer B can be concatenated to provide a 32-bit counting range. Note that the prescaler can only be used when the timers are used individually.

The available modes for each GPTM block are shown in Table 10-4 on page 483.

Table 10-4. General-Purpose Timer Capabilities

Mode	Timer Use	Count Direction	Counter Size	Prescaler Size ^a
One-shot	Individual	Up or Down	16-bit	8-bit
	Concatenated	Up or Down	32-bit	-
Periodic	Individual	Up or Down	16-bit	8-bit
	Concatenated	Up or Down	32-bit	-
RTC	Concatenated	Up	32-bit	-
Edge Count	Individual	Down	16-bit	8-bit
Edge Time	Individual	Down	16-bit	-
PWM	Individual	Down	16-bit	-

a. The prescaler is only available when the timers are used individually

Software configures the GPTM using the **GPTM Configuration (GPTMCFG)** register (see page 495), the **GPTM Timer A Mode (GPTMTAMR)** register (see page 496), and the **GPTM Timer B Mode (GPTMTBMR)** register (see page 498). When in one of the concatenated modes, Timer A and Timer B can only operate in one mode. However, when configured in an individual mode, Timer A and Timer B can be independently configured in any combination of the individual modes.

10.3.1 GPTM Reset Conditions

After reset has been applied to the GPTM module, the module is in an inactive state, and all control registers are cleared and in their default states. Counters Timer A and Timer B are initialized to all 1s, along with their corresponding load registers: the **GPTM Timer A Interval Load (GPTMTAILR)** register (see page 513) and the **GPTM Timer B Interval Load (GPTMTBILR)** register (see page 514) and shadow registers: the **GPTM Timer A Value (GPTMTAV)** register (see page 523) and the **GPTM**

Timer B Value (GPTMTBV) register (see page 524). The prescale counters are initialized to 0x00: the **GPTM Timer A Prescale (GPTMTAPR)** register (see page 517) and the **GPTM Timer B Prescale (GPTMTBPR)** register (see page 518).

10.3.2 Timer Modes

This section describes the operation of the various timer modes. When using Timer A and Timer B in concatenated mode, only the Timer A control and status bits must be used; there is no need to use Timer B control and status bits. The GPTM is placed into individual mode by writing a value of 0x4 to the **GPTM Configuration (GPTMCFG)** register (see page 495). In the following sections, the variable "n" is used in bit field and register names to imply either a Timer A function or a Timer B function. The prescaler is only available in the 16-bit one-shot, periodic, and input edge count timer mode. Throughout this section, the timeout event in down-count mode is 0x0 and in up-count mode is the value in the **GPTM Timer n Match (GPTMTnMATCH)** and the optional **GPTM Timer n Prescale Match (GPTMTnPMR)** registers.

10.3.2.1 One-Shot/Periodic Timer Mode

The selection of one-shot or periodic mode is determined by the value written to the $TnMR$ field of the **GPTM Timer n Mode (GPTMTnMR)** register (see page 496). The timer is configured to count up or down using the $TnCDIR$ bit in the **GPTMTnMR** register.

When software sets the $TnEN$ bit in the **GPTM Control (GPTMCTL)** register (see page 500), the timer begins counting up from 0x0 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 a trigger to begin counting (see the section called "Wait-for-Trigger Mode" on page 485).

When the timer is counting down and it reaches the timeout event (0x0), the timer reloads its start value from the **GPTMTnILR** and the **GPTMTnPR** registers on the next cycle. When the timer is counting up and it reaches the timeout event (the value in the **GPTMTnILR** and the **GPTMTnPR** registers), the timer reloads with 0x0. 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, the timer starts counting again on the next cycle. In periodic, snap-shot mode ($TnSNAPS$ bit in the **GPTMTnMR** register is set), the actual free-running value of the timer at the time-out event is loaded into the **GPTMTnR** 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 $TnTORIS$ bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register (see page 505), and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register (see page 511). If the timeout interrupt is enabled in the **GPTM Interrupt Mask (GPTMIMR)** register (see page 503), the GPTM also sets the $TnTOMIS$ bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register (see page 508). By setting the $TnMIE$ bit in the **GPTMTAMR** register, an interrupt can also be generated when the Timer value equals the value loaded into the **GPTM Timer n Match (GPTMTnMATCH)** and **GPTM Timer n Prescale Match (GPTMTnPMR)** registers. This interrupt has the same status, masking, and clearing functions as the timeout interrupt. The ADC trigger is enabled by setting the $TnOTE$ bit in **GPTMCTL**. The μ DMA trigger is enabled by configuring and enabling the appropriate μ DMA channel. See "Channel Configuration" on page 369.

If software updates the **GPTMTnILR** register while the counter is counting down, the counter loads the new value on the next clock cycle and continues counting down from the new value. If software updates the **GPTMTnILR** register while the counter is counting up, the timeout event is changed on the next cycle to the new value. If software updates the **GPTM Timer n Value (GPTMTnV)** register while the counter is counting up or down, the counter loads the new value on the next clock cycle and continues counting from the new value. If software updates the **GPTMTnMATCHR** register

while the counter is counting, the counter loads the new value on the next clock cycle and continues counting from the new value.

If the `TnSTALL` bit in the **GPTMCTL** register is set, the timer freezes counting while the processor is halted by the debugger. The timer resumes counting when the processor resumes execution.

The following table 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 10-5. 16-Bit Timer With Prescaler Configurations

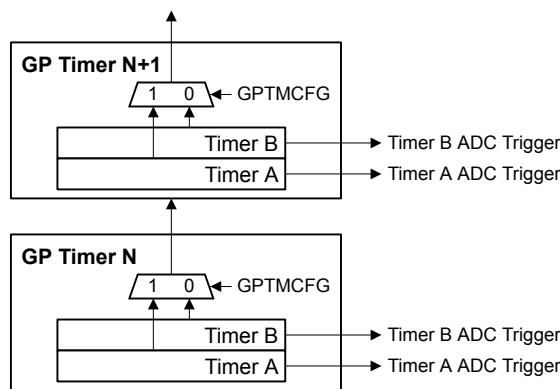
Prescale	#Clock (T_c) ^a	Max Time	Units
00000000	1	0.8192	μs
00000001	2	1.6384	μs
00000010	3	2.4576	μs
-----	--	--	--
11111101	254	208.0768	μs
11111110	255	208.896	μs
11111111	256	209.7152	μs

a. T_c is the clock period.

Wait-for-Trigger Mode

The Wait-for-Trigger mode allows daisy chaining of the timer modules such that once configured, a single timer can initiate multiple timing events using the Timer triggers. Wait-for-Trigger mode is enabled by setting the `TnWOT` bit in the **GPTMTnMR** register. When the `TnWOT` bit is set, Timer N+1 does not begin counting until the timer in the previous position in the daisy chain (Timer N) reaches its time-out event. The daisy chain is configured such that GPTM1 always follows GPTM0, GPTM2 follows GPTM1, and so on. If Timer A is in 32-bit mode (controlled by the **GPTMCFG** bit in the **GPTMCFG** register), it triggers Timer A in the next module. If Timer A is in 16-bit mode, it triggers Timer B in the same module, and Timer B triggers Timer A in the next module. Care must be taken that the `TAWOT` bit is never set in GPTM0. Figure 10-2 on page 485 shows how the **GPTMCFG** bit affects the daisy chain. This function is valid for both one-shot and periodic modes.

Figure 10-2. Timer Daisy Chain



10.3.2.2 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 an up-counter. When RTC mode is selected for the first time after reset, the

counter is loaded with a value of 0x1. All subsequent load values must be written to the **GPTM Timer A Interval Load (GPTMTAILR)** register (see page 513).

The input clock on an even CCP input is required to be 32.768 KHz in RTC mode. The clock signal is then divided down to a 1-Hz rate and is passed along to the input of the counter.

When software writes the **TAEN** bit in the **GPTMCTL** register, the counter starts counting up from its preloaded value of 0x1. When the current count value matches the preloaded value in the **GPTMTAMATCHR** register, the GPTM asserts the **RTCRIS** bit in **GPTMRIS** and continues counting until either a hardware reset, or it is disabled by software (clearing the **TAEN** bit). When the timer value reaches the terminal count, the timer rolls over and continues counting up from 0x0. If the RTC interrupt is enabled in **GPTMIMR**, the GPTM also sets the **RTCMIS** bit in **GPTMMIS** and generates a controller interrupt. The status flags are cleared by writing the **RTCCINT** bit in **GPTMICR**.

In addition to generating interrupts, a μDMA trigger can be generated. The μDMA trigger is enabled by configuring and enabling the appropriate μDMA channel. See “Channel Configuration” on page 369.

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

10.3.2.3 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 24-bit down-counter including the optional prescaler with the upper count value stored in the **GPTM Timer n Prescale (GPTMTnPR)** register and the lower bits in the **GPTMTnILR** register. 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 **TnCMR** bit of the **GPTMTnMR** register must be cleared. The type of edge that the timer counts is determined by the **TnEVENT** fields of the **GPTMCTL** register. During initialization, the **GPTMTnMATCHR** and **GPTMTnPMR** registers are configured so that the difference between the value in the **GPTMTnILR** and **GPTMTnPR** registers and the **GPTMTnMATCHR** and **GPTMTnPMR** registers equals the number of edge events that must be counted.

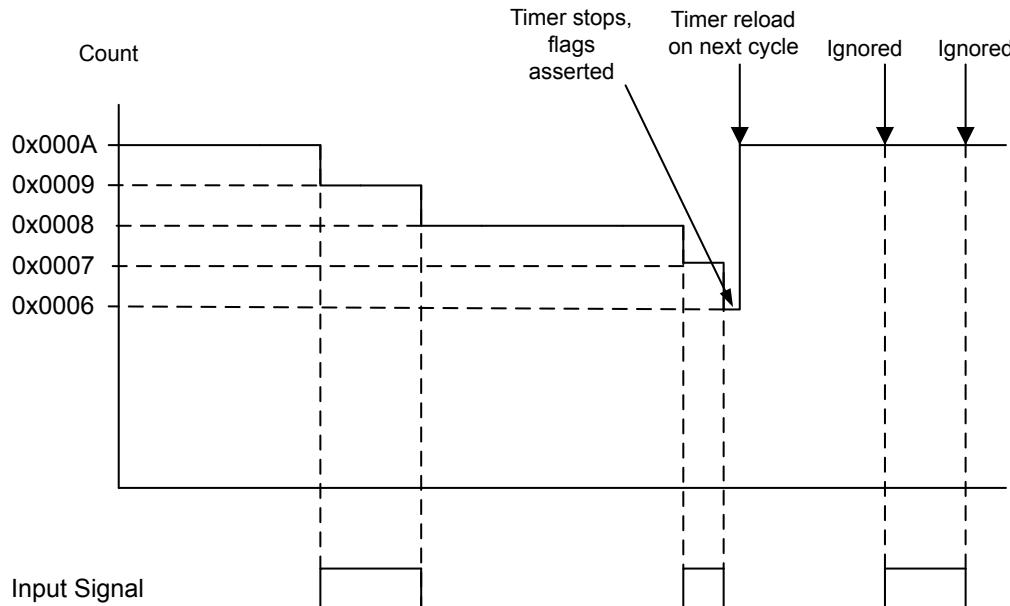
When software writes the **TnEN** bit in the **GPTM Control (GPTMCTL)** register, the timer is enabled for event capture. Each input event on the CCP pin decrements the counter by 1 until the event count matches **GPTMTnMATCHR** and **GPTMTnPMR**. When the counts match, the GPTM asserts the **CnMRIS** bit in the **GPTMRIS** register (and the **CnMMIS** bit, if the interrupt is not masked).

In addition to generating interrupts, an ADC and/or a μDMA trigger can be generated. The ADC trigger is enabled by setting the **TnOTE** bit in **GPTMCTL**. The μDMA trigger is enabled by configuring and enabling the appropriate μDMA channel. See “Channel Configuration” on page 369.

After the match value is reached, the counter is then reloaded using the value in **GPTMTnILR** and **GPTMTnPR** registers, and stopped because the GPTM automatically clears the **TnEN** bit in the **GPTMCTL** register. Once the event count has been reached, all further events are ignored until **TnEN** is re-enabled by software.

Figure 10-3 on page 487 shows how Input Edge-Count mode works. In this case, the timer start value is set to **GPTMTnILR** =0x000A and the match value is set to **GPTMTnMATCHR** =0x0006 so that four edge events are counted. The counter is configured to detect both edges of the input signal.

Note that the last two edges are not counted because the timer automatically clears the **TnEN** bit after the current count matches the value in the **GPTMTnMATCHR** register.

Figure 10-3. Input Edge-Count Mode Example

10.3.2.4 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.

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

In Edge-Time mode, the timer is configured as a 16-bit down-counter. In this mode, the timer is initialized to the value loaded in the **GPTMTnILR** register. The timer is capable of capturing three types of events: rising edge, falling edge, or both. The timer is placed into Edge-Time mode by setting the **TnCMR** bit in the **GPTMTnMR** register, and the type of event that the timer captures is determined by the **TnEVENT** fields of the **GPTMCTL** register.

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

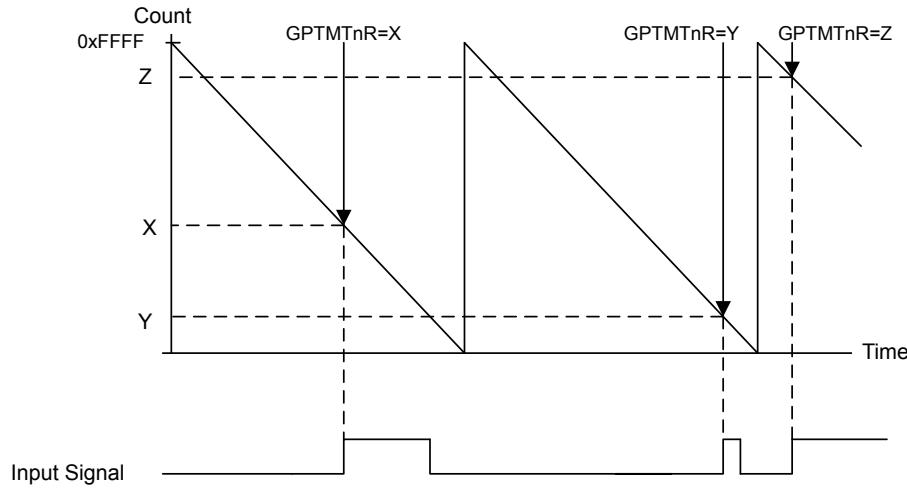
In addition to generating interrupts, an ADC and/or a μDMA trigger can be generated. The ADC trigger is enabled by setting the **TnOTE** bit in **GPTMCTL**. The μDMA trigger is enabled by configuring and enabling the appropriate μDMA channel. See “Channel Configuration” on page 369.

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 timeout value, it is reloaded with the value from the **GPTMTnILR** register.

Figure 10-4 on page 488 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 the **GPTMTnR** register).

Figure 10-4. 16-Bit Input Edge-Time Mode Example



10.3.2.5 PWM Mode

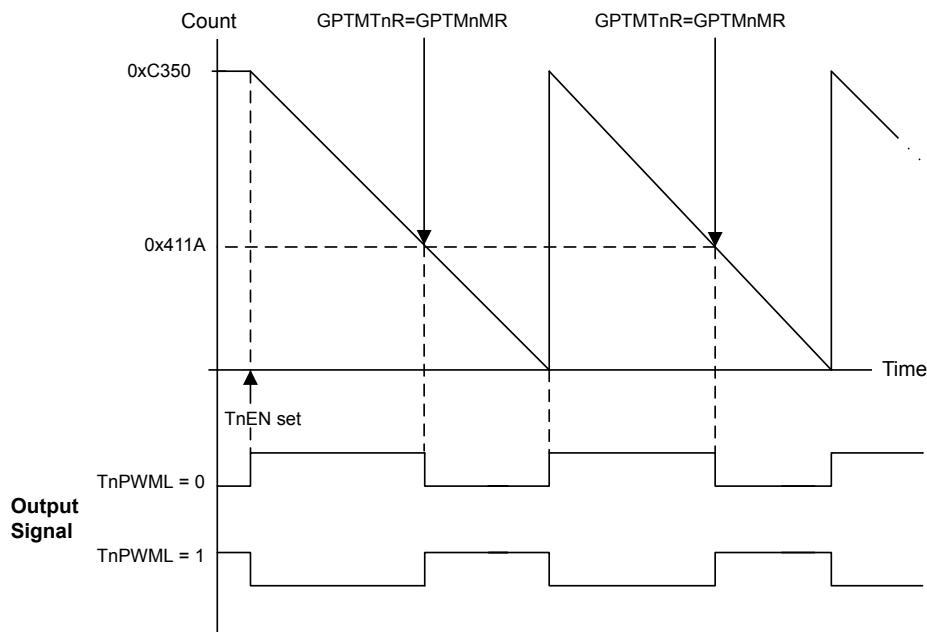
Note: The prescaler is not available in 16-Bit PWM mode.

The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a 16-bit down-counter with a start value (and thus period) defined by the **GPTMTnILR** register. In this mode, the PWM frequency and period are synchronous events and therefore guaranteed to be glitch free. PWM mode is enabled with the **GPTMTnMR** register by setting the **TnAMS** bit to 0x1, the **TnCMR** bit to 0x0, and the **TnMR** field to 0x2.

When software writes the **TnEN** bit in the **GPTMCTL** register, the counter begins counting down until it reaches the 0x0 state. On the next counter cycle, the counter reloads its start value from the **GPTMTnILR** register 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 **GPTMTnMATCHR** register. Software has the capability of inverting the output PWM signal by setting the **TnPWML** bit in the **GPTMCTL** register.

Figure 10-5 on page 489 shows how to generate an output PWM with a 1-ms period and a 66% duty cycle assuming a 50-MHz input clock and **TnPWML**=0 (duty cycle would be 33% for the **TnPWML**=1 configuration). For this example, the start value is **GPTMTnILR**=0xC350 and the match value is **GPTMTnMATCHR**=0x411A.

Figure 10-5. 16-Bit PWM Mode Example

10.3.3 DMA Operation

The timers each have a dedicated µDMA channel and can provide a request signal to the µDMA controller. The request is a burst type and occurs whenever a timer raw interrupt condition occurs. The arbitration size of the µDMA 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 µDMA transfer for a total of 256 items, with a burst size of 8 items. Each time the timer times out, the µDMA controller transfers 8 items, until all 256 items have been transferred.

No other special steps are needed to enable Timers for µDMA operation. Refer to “Micro Direct Memory Access (µDMA)” on page 365 for more details about programming the µDMA controller.

10.3.4 Accessing Concatenated Register Values

The GPTM is placed into concatenated mode by writing a 0x0 or a 0x1 to the GPTMCFG bit field in the **GPTM Configuration (GPTMCFG)** register. In both configurations, certain registers are concatenated to form pseudo 32-bit registers. These registers include:

- **GPTM Timer A Interval Load (GPTMTAILR)** register [15:0], see page 513
- **GPTM Timer B Interval Load (GPTMTBILR)** register [15:0], see page 514
- **GPTM Timer A (GPTMTAR)** register [15:0], see page 521
- **GPTM Timer B (GPTMTBR)** register [15:0], see page 522
- **GPTM Timer A Value (GPTMTAV)** register [15:0], see page 523

- **GPTM Timer B Value (GPTMTBV)** register [15:0], see page 524
- **GPTM Timer A Match (GPTMTAMATCHR)** register [15:0], see page 515
- **GPTM Timer B Match (GPTMTBMATCHR)** register [15:0], see page 516

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 32-bit read access to **GPTMTAR** returns the value:

`GPTMTBR[15:0]:GPTMTAR[15:0]`

A 32-bit read access to **GPTMTAV** returns the value:

`GPTMTBV[15:0]:GPTMTAV[15:0]`

10.4 Initialization and Configuration

To use a GPTM, the appropriate `TIMERn` bit must be set in the **RCGC1** register (see page 276). If using any CCP pins, the clock to the appropriate GPIO module must be enabled via the **RCGC1** register (see page 276). To find out which GPIO port to enable, refer to Table 24-4 on page 1182. Configure the `PMCn` fields in the **GPIOCTL** register to assign the CCP signals to the appropriate pins (see page 465 and Table 24-5 on page 1190).

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

10.4.1 32-Bit One-Shot/Periodic Timer Mode

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

1. Ensure the timer is disabled (the `TAEN` bit in the **GPTMCTL** register is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of `0x0000.0000`.
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)**.

If the **TAMIE** bit in the **GPTMTAMR** register is set, the **RTCRIS** bit in the **GPTMRIS** register is set, and the timer continues counting. In One-Shot mode, the timer stops counting after the time-out event. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode reloads the timer and continues counting after the time-out event.

10.4.2 32-Bit Real-Time Clock (RTC) Mode

To use the RTC mode, the timer must have a 32.768-KHz input signal on an even CCP input. To enable the RTC feature, follow these steps:

1. Ensure the timer is disabled (the **TAEN** bit is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0000.0001.
3. Write the match value to the **GPTM Timer A Match Register (GPTMTAMATCHR)**.
4. Set/clear the **RTCEN** bit in the **GPTM Control Register (GPTMCTL)** as needed.
5. If interrupts are required, set the **RTCIM** bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.
6. Set the **TAEN** bit in the **GPTMCTL** register to enable the timer and start counting.

When the timer count equals the value in the **GPTMTAMATCHR** register, the GPTM asserts the **RTCRIS** bit in the **GPTMRIS** register and continues counting until Timer A is disabled or a hardware reset. The interrupt is cleared by writing the **RTCCINT** bit in the **GPTMICR** register.

10.4.3 16-Bit One-Shot/Periodic Timer Mode

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

1. Ensure the timer is disabled (the **TnEN** bit is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0000.0004.
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)**.

If the TnMIE bit in the **GPTMTnMR** register is set, the RTCRIS bit in the **GPTMRIS** register is set, and the timer continues counting. In One-Shot mode, the timer stops counting after the time-out event. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode reloads the timer and continues counting after the time-out event.

10.4.4 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 0x0000.0004.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the TnCMR field to 0x0 and the TnMR field to 0x3.
4. Configure the type of event(s) that the timer captures by writing the TnEVENT field of the **GPTM Control (GPTMCTL)** register.
5. 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 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 programmed 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 492 through step 9 on page 492.

10.4.5 16-Bit Input Edge Timing Mode

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

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x0000.0004.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the TnCMR field to 0x1 and the TnMR field to 0x3.

4. Configure the type of event that the timer captures by writing the TnEVENT field of the **GPTM Control (GPTMCTL)** register.
5. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
6. If interrupts are required, set the CnEIM bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
7. Set the TnEN bit in the **GPTM Control (GPTMCTL)** register to enable the timer and start counting.
8. Poll the CnERIS bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the CnECINT bit of the **GPTM Interrupt Clear (GPTMICR)** register. The time at which the event happened can be obtained by reading the **GPTM 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.

10.4.6 16-Bit PWM Mode

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

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x0000.0004.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, set the TnAMS bit to 0x1, the TnCMR bit to 0x0, and the TnMR field to 0x2.
4. Configure the output state of the PWM signal (whether or not it is inverted) in the TnPWL field of the **GPTM Control (GPTMCTL)** register.
5. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
6. Load the **GPTM Timer n Match (GPTMTnMATCHR)** register with the match 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.

10.5 Register Map

Table 10-6 on page 494 lists the GPTM registers. The offset listed is a hexadecimal increment to the register's address, relative to that timer's base address:

- Timer 0: 0x4003.0000
- Timer 1: 0x4003.1000
- Timer 2: 0x4003.2000
- Timer 3: 0x4003.3000

Note that the GP Timer module clock must be enabled before the registers can be programmed (see page 276).

Table 10-6. Timers Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPTMCFG	R/W	0x0000.0000	GPTM Configuration	495
0x004	GPTMTAMR	R/W	0x0000.0000	GPTM Timer A Mode	496
0x008	GPTMTBMR	R/W	0x0000.0000	GPTM Timer B Mode	498
0x00C	GPTMCTL	R/W	0x0000.0000	GPTM Control	500
0x018	GPTMIMR	R/W	0x0000.0000	GPTM Interrupt Mask	503
0x01C	GPTMRIS	RO	0x0000.0000	GPTM Raw Interrupt Status	505
0x020	GPTMMIS	RO	0x0000.0000	GPTM Masked Interrupt Status	508
0x024	GPTMICR	W1C	0x0000.0000	GPTM Interrupt Clear	511
0x028	GPTMTAILR	R/W	0xFFFF.FFFF	GPTM Timer A Interval Load	513
0x02C	GPTMTBILR	R/W	0x0000.FFFF	GPTM Timer B Interval Load	514
0x030	GPTMTAMATCHR	R/W	0xFFFF.FFFF	GPTM Timer A Match	515
0x034	GPTMTBMATCHR	R/W	0x0000.FFFF	GPTM Timer B Match	516
0x038	GPTMTAPR	R/W	0x0000.0000	GPTM Timer A Prescale	517
0x03C	GPTMTBPR	R/W	0x0000.0000	GPTM Timer B Prescale	518
0x040	GPTMTAPMR	R/W	0x0000.0000	GPTM TimerA Prescale Match	519
0x044	GPTMTBPMR	R/W	0x0000.0000	GPTM TimerB Prescale Match	520
0x048	GPTMTAR	RO	0xFFFF.FFFF	GPTM Timer A	521
0x04C	GPTMTBR	RO	0x0000.FFFF	GPTM Timer B	522
0x050	GPTMTAV	RW	0xFFFF.FFFF	GPTM Timer A Value	523
0x054	GPTMTBV	RW	0x0000.FFFF	GPTM Timer B Value	524

10.6 Register Descriptions

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

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

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

Important: Bits in this register should only be changed when the TAEN and TBEN bits in the **GPTMCTL** register are cleared.

GPTM Configuration (GPTMCFG)

Timer 0 base: 0x4003.0000

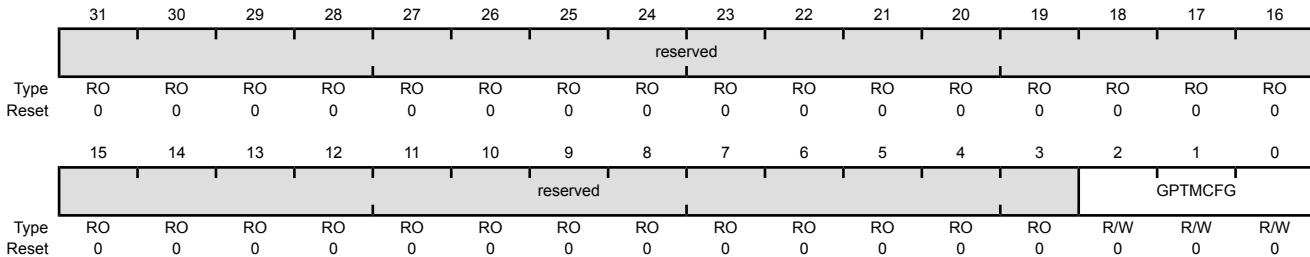
Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 base: 0x4003.3000

Offset 0x000

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description												
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.												
2:0	GPTMCFG	R/W	0x0	<p>GPTM Configuration</p> <p>The GPTMCFG 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>32-bit timer configuration.</td> </tr> <tr> <td>0x1</td> <td>32-bit real-time clock (RTC) counter configuration.</td> </tr> <tr> <td>0x2-0x3</td> <td>Reserved</td> </tr> <tr> <td>0x4</td> <td>16-bit timer configuration. The function is controlled by bits 1:0 of GPTMTAMR and GPTMTBMR.</td> </tr> <tr> <td>0x5-0x7</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	32-bit timer configuration.	0x1	32-bit real-time clock (RTC) counter configuration.	0x2-0x3	Reserved	0x4	16-bit timer configuration. The function is controlled by bits 1:0 of GPTMTAMR and GPTMTBMR .	0x5-0x7	Reserved
Value	Description															
0x0	32-bit timer configuration.															
0x1	32-bit real-time clock (RTC) counter configuration.															
0x2-0x3	Reserved															
0x4	16-bit timer configuration. The function is controlled by bits 1:0 of GPTMTAMR and GPTMTBMR .															
0x5-0x7	Reserved															

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 PWM mode, set the TAAMS bit, clear the TACMR bit, and configure the TAMR field to 0x2.

This register controls the modes for Timer A when it is used individually. When Timer A and Timer B are concatenated, this register controls the modes for both Timer A and Timer B, and the contents of **GPTMTBMR** are ignored.

Important: Bits in this register should only be changed when the TAEN bit in the **GPTMCTL** register is cleared.

GPTM Timer A Mode (GPTMTAMR)

Timer 0 base: 0x4003.0000

Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 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							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	R/W														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	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	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 GPTM Timer A (GPTMTAR) 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 (TAEN is set in the GPTMCTL register), Timer A does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain, see Figure 10-2 on page 485. 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> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The match interrupt is disabled.</td></tr> <tr> <td>1</td><td>An interrupt is generated when the match value in the GPTMTAMATCHR register is reached in the one-shot and periodic modes.</td></tr> </tbody> </table>	Value	Description	0	The match interrupt is disabled.	1	An interrupt is generated when the match value in the GPTMTAMATCHR register is reached in the one-shot and periodic modes.				
Value	Description													
0	The match interrupt is disabled.													
1	An interrupt is generated when the match value in the GPTMTAMATCHR register is reached in the one-shot and periodic modes.													
4	TACDIR	R/W	0	<p>GPTM Timer A Count Direction</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The timer counts down.</td></tr> <tr> <td>1</td><td>When in one-shot or periodic mode, the timer counts up. When counting up, the timer starts from a value of 0x0.</td></tr> </tbody> </table> <p>When in PWM or RTC mode, the status of this bit is ignored. PWM mode always counts down and RTC mode always counts up.</p>	Value	Description	0	The timer counts down.	1	When in one-shot or periodic mode, the timer counts up. When counting up, the timer starts from a value of 0x0.				
Value	Description													
0	The timer counts down.													
1	When in one-shot or periodic mode, the timer counts up. When counting up, the timer starts from a value of 0x0.													
3	TAAMS	R/W	0	<p>GPTM Timer A Alternate Mode Select</p> <p>The TAAMS values are defined as follows:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Capture mode is enabled.</td></tr> <tr> <td>1</td><td>PWM mode is enabled.</td></tr> </tbody> </table> <p>Note: To enable PWM mode, you must also clear the TACMR bit and configure the TAMR field to 0x2.</p>	Value	Description	0	Capture mode is enabled.	1	PWM mode is enabled.				
Value	Description													
0	Capture mode is enabled.													
1	PWM mode is enabled.													
2	TACMR	R/W	0	<p>GPTM Timer A Capture Mode</p> <p>The TACMR values are defined as follows:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Edge-Count mode</td></tr> <tr> <td>1</td><td>Edge-Time mode</td></tr> </tbody> </table>	Value	Description	0	Edge-Count mode	1	Edge-Time mode				
Value	Description													
0	Edge-Count mode													
1	Edge-Time mode													
1:0	TAMR	R/W	0x0	<p>GPTM Timer A Mode</p> <p>The TAMR values are defined as follows:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Reserved</td></tr> <tr> <td>0x1</td><td>One-Shot Timer mode</td></tr> <tr> <td>0x2</td><td>Periodic Timer mode</td></tr> <tr> <td>0x3</td><td>Capture mode</td></tr> </tbody> </table> <p>The Timer mode is based on the timer configuration defined by bits 2:0 in the GPTMCFG register.</p>	Value	Description	0x0	Reserved	0x1	One-Shot Timer mode	0x2	Periodic Timer mode	0x3	Capture mode
Value	Description													
0x0	Reserved													
0x1	One-Shot Timer mode													
0x2	Periodic Timer mode													
0x3	Capture mode													

Register 3: GPTM Timer B Mode (GPTMTBMR), offset 0x008

This register configures the GPTM based on the configuration selected in the **GPTMCFG** register. When in PWM mode, set the TBAMS bit, clear the TBCMR bit, and configure the TBMR field to 0x2.

This register controls the modes for Timer B when it is used individually. When Timer A and Timer B are concatenated, this register is ignored and **GPTMTBMR** controls the modes for both Timer A and Timer B.

Important: Bits in this register should only be changed when the TBEN bit in the **GPTMCTL** register is cleared.

GPTM Timer B Mode (GPTMTBMR)

Timer 0 base: 0x4003.0000

Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 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							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	R/W														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	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	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 GPTM Timer B (GPTMTBR) 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 (TBEN is set in the GPTMCTL register), Timer B does not begin counting until it receives an trigger from the timer in the previous position in the daisy chain. See Figure 10-2 on page 485. 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> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The match interrupt is disabled.</td></tr> <tr> <td>1</td><td>An interrupt is generated when the match value in the GPTMTBMATCHR register is reached in the one-shot and periodic modes.</td></tr> </tbody> </table>	Value	Description	0	The match interrupt is disabled.	1	An interrupt is generated when the match value in the GPTMTBMATCHR register is reached in the one-shot and periodic modes.				
Value	Description													
0	The match interrupt is disabled.													
1	An interrupt is generated when the match value in the GPTMTBMATCHR register is reached in the one-shot and periodic modes.													
4	TBCDIR	R/W	0	<p>GPTM Timer B Count Direction</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The timer counts down.</td></tr> <tr> <td>1</td><td>When in one-shot or periodic mode, the timer counts up. When counting up, the timer starts from a value of 0x0.</td></tr> </tbody> </table> <p>When in PWM or RTC mode, the status of this bit is ignored. PWM mode always counts down and RTC mode always counts up.</p>	Value	Description	0	The timer counts down.	1	When in one-shot or periodic mode, the timer counts up. When counting up, the timer starts from a value of 0x0.				
Value	Description													
0	The timer counts down.													
1	When in one-shot or periodic mode, the timer counts up. When counting up, the timer starts from a value of 0x0.													
3	TBAMS	R/W	0	<p>GPTM Timer B Alternate Mode Select</p> <p>The TBAMS values are defined as follows:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Capture mode is enabled.</td></tr> <tr> <td>1</td><td>PWM mode is enabled.</td></tr> </tbody> </table> <p>Note: To enable PWM mode, you must also clear the TBCMR bit and set the TBMR field to 0x2.</p>	Value	Description	0	Capture mode is enabled.	1	PWM mode is enabled.				
Value	Description													
0	Capture mode is enabled.													
1	PWM mode is enabled.													
2	TBCMR	R/W	0	<p>GPTM Timer B Capture Mode</p> <p>The TBCMR values are defined as follows:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Edge-Count mode</td></tr> <tr> <td>1</td><td>Edge-Time mode</td></tr> </tbody> </table>	Value	Description	0	Edge-Count mode	1	Edge-Time mode				
Value	Description													
0	Edge-Count mode													
1	Edge-Time mode													
1:0	TBMR	R/W	0x0	<p>GPTM Timer B Mode</p> <p>The TBMR values are defined as follows:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Reserved</td></tr> <tr> <td>0x1</td><td>One-Shot Timer mode</td></tr> <tr> <td>0x2</td><td>Periodic Timer mode</td></tr> <tr> <td>0x3</td><td>Capture mode</td></tr> </tbody> </table> <p>The timer mode is based on the timer configuration defined by bits 2:0 in the GPTMCFG register.</p>	Value	Description	0x0	Reserved	0x1	One-Shot Timer mode	0x2	Periodic Timer mode	0x3	Capture mode
Value	Description													
0x0	Reserved													
0x1	One-Shot Timer mode													
0x2	Periodic Timer mode													
0x3	Capture mode													

Register 4: GPTM Control (GPTMCTL), offset 0x00C

This register is used alongside the **GPTMCFG** and **GMTMTnMR** registers to fine-tune the timer configuration, and to enable other features such as timer stall and the output trigger. The output trigger can be used to initiate transfers on the ADC module.

Important: Bits in this register should only be changed when the **TnEN** bit for the respective timer is cleared.

GPTM Control (GPTMCTL)

Timer 0 base: 0x4003.0000

Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 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	RTCN	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	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.
14	TBPWML	R/W	0	GPTM Timer B PWM Output Level The TBPWML values are defined as follows: Value Description 0 Output is unaffected. 1 Output is inverted.
13	TBOTE	R/W	0	GPTM Timer B Output Trigger Enable The TBOTE values are defined as follows: Value Description 0 The output Timer B ADC trigger is disabled. 1 The output Timer B ADC trigger is enabled. In addition, the ADC must be enabled and the timer selected as a trigger source with the EMn bit in the ADCCEMUX register (see page 581).
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
11:10	TBEVENT	R/W	0x0	GPTM Timer B Event Mode The TBEVENT values are defined as follows: Value Description 0x0 Positive edge 0x1 Negative edge 0x2 Reserved 0x3 Both edges
9	TBSTALL	R/W	0	GPTM Timer B Stall Enable The TBSTALL values are defined as follows: Value Description 0 Timer B continues counting while the processor is halted by the debugger. 1 Timer B freezes counting while the processor is halted by the debugger. If the processor is executing normally, the TBSTALL bit is ignored.
8	TBEN	R/W	0	GPTM Timer B Enable The TBEN values are defined as follows: Value Description 0 Timer B is disabled. 1 Timer B is enabled and begins counting or the capture logic is enabled based on the GPTMCFG register.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	TAPWML	R/W	0	GPTM Timer A PWM Output Level The TAPWML values are defined as follows: Value Description 0 Output is unaffected. 1 Output is inverted.
5	TAOTE	R/W	0	GPTM Timer A Output Trigger Enable The TAOTE values are defined as follows: Value Description 0 The output Timer A ADC trigger is disabled. 1 The output Timer A ADC trigger is enabled. In addition, the ADC must be enabled and the timer selected as a trigger source with the EMn bit in the ADCEMUX register (see page 581).

Bit/Field	Name	Type	Reset	Description										
4	RTCEN	R/W	0	<p>GPTM RTC Enable</p> <p>The RTCEN values are defined as follows:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>RTC counting is disabled.</td></tr> <tr> <td>1</td><td>RTC counting is enabled.</td></tr> </tbody> </table>	Value	Description	0	RTC counting is disabled.	1	RTC counting is enabled.				
Value	Description													
0	RTC counting is disabled.													
1	RTC counting is enabled.													
3:2	TAEVENT	R/W	0x0	<p>GPTM Timer A Event Mode</p> <p>The TAEVENT values are defined as follows:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Positive edge</td></tr> <tr> <td>0x1</td><td>Negative edge</td></tr> <tr> <td>0x2</td><td>Reserved</td></tr> <tr> <td>0x3</td><td>Both edges</td></tr> </tbody> </table>	Value	Description	0x0	Positive edge	0x1	Negative edge	0x2	Reserved	0x3	Both edges
Value	Description													
0x0	Positive edge													
0x1	Negative edge													
0x2	Reserved													
0x3	Both edges													
1	TASTALL	R/W	0	<p>GPTM Timer A Stall Enable</p> <p>The TASTALL values are defined as follows:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Timer A continues counting while the processor is halted by the debugger.</td></tr> <tr> <td>1</td><td>Timer A freezes counting while the processor is halted by the debugger.</td></tr> </tbody> </table> <p>If the processor is executing normally, the TASTALL bit is ignored.</p>	Value	Description	0	Timer A continues counting while the processor is halted by the debugger.	1	Timer A freezes counting while the processor is halted by the debugger.				
Value	Description													
0	Timer A continues counting while the processor is halted by the debugger.													
1	Timer A freezes counting while the processor is halted by the debugger.													
0	TAEN	R/W	0	<p>GPTM Timer A Enable</p> <p>The TAEN values are defined as follows:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Timer A 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 GPTMCFG 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 GPTMCFG 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 GPTMCFG register.													

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

This register allows software to enable/disable GPTM controller-level interrupts. Setting a bit enables the corresponding interrupt, while clearing a bit disables it.

GPTM Interrupt Mask (GPTMIMR)

Timer 0 base: 0x4003.0000

Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 base: 0x4003.3000

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				TBMIM	CBEIM	CBMIM	TBTOM	reserved			TAMIM	RTCIM	CAEIM	CAMIM	TATOIM
Type	RO	RO	RO	RO	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
31:12	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.
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	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	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 **GPTMMIR** register. Each bit can be cleared by writing a 1 to its corresponding bit in **GPTMICR**.

GPTM Raw Interrupt Status (GPTMRIS)

Timer 0 base: 0x4003.0000

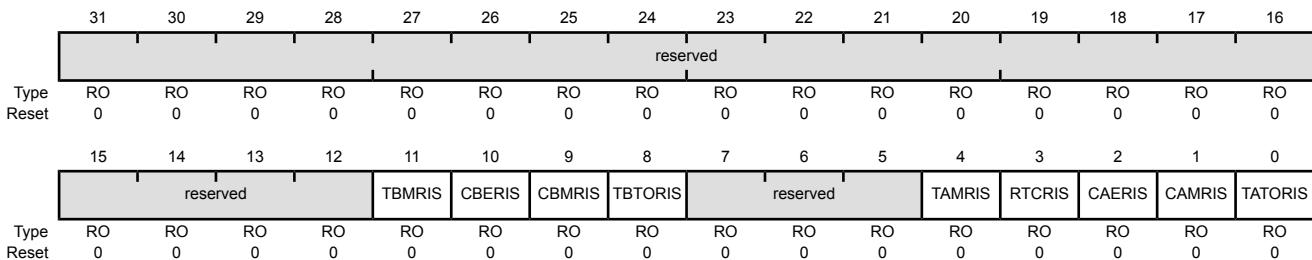
Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 base: 0x4003.3000

Offset 0x01C

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	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.
11	TBMRIS	RO	0	GPTM Timer B Mode Match Raw Interrupt
		Value	Description	
		1	The TBMIE bit is set in the GPTMTBMR register, and the match value in the GPTMTBMATCHR register has been reached when in the one-shot and periodic modes.	
		0	The match value has not been reached.	
				This bit is cleared by writing a 1 to the TBMCINT bit in the GPTMICR register.
10	CBERIS	RO	0	GPTM Capture B Event Raw Interrupt
		Value	Description	
		1	The Capture B event has occurred.	
		0	The Capture B event has not occurred.	
				This bit is cleared by writing a 1 to the CBECINT bit in the GPTMICR register.
9	CBMRIS	RO	0	GPTM Capture B Match Raw Interrupt
		Value	Description	
		1	The Capture B match has occurred.	
		0	The Capture B match has not occurred.	
				This bit is cleared by writing a 1 to the CBMCINT bit in the GPTMICR register.

Bit/Field	Name	Type	Reset	Description						
8	TBTORIS	RO	0	<p>GPTM Timer B Time-Out Raw Interrupt</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Timer B has timed out.</td></tr> <tr> <td>0</td><td>Timer B has not timed out.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the TBTOCINT bit in the GPTMICR register.</p>	Value	Description	1	Timer B has timed out.	0	Timer B has not timed out.
Value	Description									
1	Timer B has timed out.									
0	Timer B has not timed out.									
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	RO	0	<p>GPTM Timer A Mode Match Raw Interrupt</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The TAMIE bit is set in the GPTMTAMR register, and the match value in the GPTMTAMATCHR register has been reached when in the one-shot and periodic modes.</td></tr> <tr> <td>0</td><td>The match value has not been reached.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the TAMCINT bit in the GPTMICR register.</p>	Value	Description	1	The TAMIE bit is set in the GPTMTAMR register, and the match value in the GPTMTAMATCHR register has been reached when in the one-shot and periodic modes.	0	The match value has not been reached.
Value	Description									
1	The TAMIE bit is set in the GPTMTAMR register, and the match value in the GPTMTAMATCHR register has been reached when in the one-shot and periodic modes.									
0	The match value has not been reached.									
3	RTCRIS	RO	0	<p>GPTM RTC Raw Interrupt</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The RTC event has occurred.</td></tr> <tr> <td>0</td><td>The RTC event has not occurred.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the RTCCINT bit in the GPTMICR register.</p>	Value	Description	1	The RTC event has occurred.	0	The RTC event has not occurred.
Value	Description									
1	The RTC event has occurred.									
0	The RTC event has not occurred.									
2	CAERIS	RO	0	<p>GPTM Capture A Event Raw Interrupt</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The Capture A event has occurred.</td></tr> <tr> <td>0</td><td>The Capture A event has not occurred.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the CAECINT bit in the GPTMICR register.</p>	Value	Description	1	The Capture A event has occurred.	0	The Capture A event has not occurred.
Value	Description									
1	The Capture A event has occurred.									
0	The Capture A event has not occurred.									
1	CAMRIS	RO	0	<p>GPTM Capture A Match Raw Interrupt</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The Capture A match has occurred.</td></tr> <tr> <td>0</td><td>The Capture A match has not occurred.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the CAMCINT bit in the GPTMICR register.</p>	Value	Description	1	The Capture A match has occurred.	0	The Capture A match has not occurred.
Value	Description									
1	The Capture A match has occurred.									
0	The Capture A match has not occurred.									

Bit/Field	Name	Type	Reset	Description
0	TATORIS	RO	0	GPTM Timer A Time-Out Raw Interrupt Value Description 1 Timer A has timed out. 0 Timer A has not timed out. This bit is cleared by writing a 1 to the TATOCINT bit in the GPTMICR register.

Register 7: GPTM Masked Interrupt Status (GPTMMIS), offset 0x020

This register shows the state of the GPTM's controller-level interrupt. If an interrupt is unmasked in **GPTMIMR**, and there is an event that causes the interrupt to be asserted, the corresponding bit is set in this register. All bits are cleared by writing a 1 to the corresponding bit in **GPTMICR**.

GPTM Masked Interrupt Status (GPTMMIS)

Timer 0 base: 0x4003.0000

Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 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	RO	RO	RO	RO	RO	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
31:12	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.
11	TBMMIS	RO	0	GPTM Timer B Mode Match Masked Interrupt
		Value	Description	
		1	An unmasked Timer B Mode Match interrupt has occurred.	
		0	A Timer B Mode Match interrupt has not occurred or is masked.	
		This bit is cleared by writing a 1 to the TBMCINT bit in the GPTMICR register.		
10	CBEMIS	RO	0	GPTM Capture B Event Masked Interrupt
		Value	Description	
		1	An unmasked Capture B event interrupt has occurred.	
		0	A Capture B event interrupt has not occurred or is masked.	
		This bit is cleared by writing a 1 to the CEBCINT bit in the GPTMICR register.		

Bit/Field	Name	Type	Reset	Description						
9	CBMMIS	RO	0	<p>GPTM Capture B Match Masked Interrupt</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An unmasked Capture B Match interrupt has occurred.</td></tr> <tr> <td>0</td><td>A Capture B Mode Match interrupt has not occurred or is masked.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the <code>CBMCINT</code> bit in the GPTMICR register.</p>	Value	Description	1	An unmasked Capture B Match interrupt has occurred.	0	A Capture B Mode Match interrupt has not occurred or is masked.
Value	Description									
1	An unmasked Capture B Match interrupt has occurred.									
0	A Capture B Mode Match interrupt has not occurred or is masked.									
8	TBTOMIS	RO	0	<p>GPTM Timer B Time-Out Masked Interrupt</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An unmasked Timer B Time-Out interrupt has occurred.</td></tr> <tr> <td>0</td><td>A Timer B Time-Out interrupt has not occurred or is masked.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the <code>TBTOCINT</code> bit in the GPTMICR register.</p>	Value	Description	1	An unmasked Timer B Time-Out interrupt has occurred.	0	A Timer B Time-Out interrupt has not occurred or is masked.
Value	Description									
1	An unmasked Timer B Time-Out interrupt has occurred.									
0	A Timer B Time-Out interrupt has not occurred or is masked.									
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	RO	0	<p>GPTM Timer A Mode Match Masked Interrupt</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An unmasked Timer A Mode Match interrupt has occurred.</td></tr> <tr> <td>0</td><td>A Timer A Mode Match interrupt has not occurred or is masked.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the <code>TAMCINT</code> bit in the GPTMICR register.</p>	Value	Description	1	An unmasked Timer A Mode Match interrupt has occurred.	0	A Timer A Mode Match interrupt has not occurred or is masked.
Value	Description									
1	An unmasked Timer A Mode Match interrupt has occurred.									
0	A Timer A Mode Match interrupt has not occurred or is masked.									
3	RTCMIS	RO	0	<p>GPTM RTC Masked Interrupt</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An unmasked RTC event interrupt has occurred.</td></tr> <tr> <td>0</td><td>An RTC event interrupt has not occurred or is masked.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the <code>RTCCINT</code> bit in the GPTMICR register.</p>	Value	Description	1	An unmasked RTC event interrupt has occurred.	0	An RTC event interrupt has not occurred or is masked.
Value	Description									
1	An unmasked RTC event interrupt has occurred.									
0	An RTC event interrupt has not occurred or is masked.									
2	CAEMIS	RO	0	<p>GPTM Capture A Event Masked Interrupt</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An unmasked Capture A event interrupt has occurred.</td></tr> <tr> <td>0</td><td>A Capture A event interrupt has not occurred or is masked.</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the <code>CAECINT</code> bit in the GPTMICR register.</p>	Value	Description	1	An unmasked Capture A event interrupt has occurred.	0	A Capture A event interrupt has not occurred or is masked.
Value	Description									
1	An unmasked Capture A event interrupt has occurred.									
0	A Capture A event interrupt has not occurred or is masked.									

Bit/Field	Name	Type	Reset	Description
1	CAMMIS	RO	0	<p>GPTM Capture A Match Masked Interrupt</p> <p>Value Description</p> <p>1 An unmasked Capture A Match interrupt has occurred.</p> <p>0 A Capture A Mode Match interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the CAMCINT bit in the GPTMICR register.</p>
0	TATOMIS	RO	0	<p>GPTM Timer A Time-Out Masked Interrupt</p> <p>Value Description</p> <p>1 An unmasked Timer A Time-Out interrupt has occurred.</p> <p>0 A Timer A Time-Out interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the TATOCINT bit in the GPTMICR register.</p>

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)

Timer 0 base: 0x4003.0000

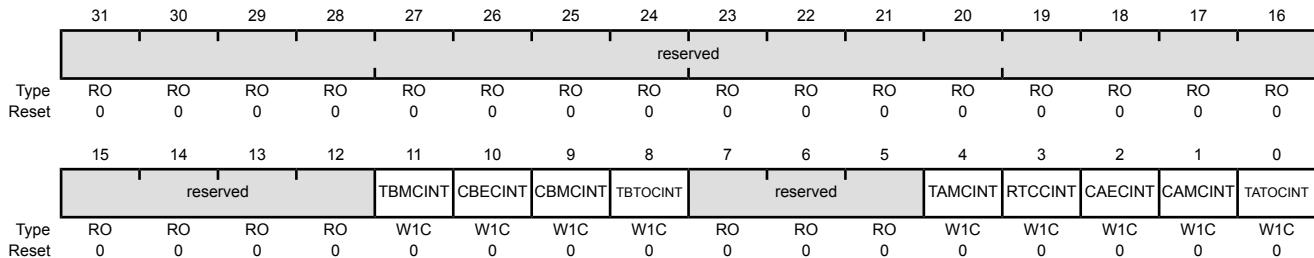
Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 base: 0x4003.3000

Offset 0x024

Type W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	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.
11	TBMCINT	W1C	0	GPTM Timer B Mode Match Interrupt Clear Writing a 1 to this bit clears the TBMRIS bit in the GPTMRIS register and the TBMMIS bit in the GPTMMIS register.
10	CBECINT	W1C	0	GPTM Capture B Event Interrupt Clear Writing a 1 to this bit clears the CBERIS bit in the GPTMRIS register and the CBEMIS bit in the GPTMMIS register.
9	CBMCINT	W1C	0	GPTM Capture B Match Interrupt Clear Writing a 1 to this bit clears the CBMRIS bit in the GPTMRIS register and the CBMMIS bit in the GPTMMIS register.
8	TBTOCINT	W1C	0	GPTM Timer B Time-Out Interrupt Clear Writing a 1 to this bit clears the TBTORIS bit in the GPTMRIS register and the TBTOMIS bit in the GPTMMIS register.
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	GPTM Timer A Mode Match Interrupt Clear Writing a 1 to this bit clears the TAMRIS bit in the GPTMRIS register and the TAMMIS bit in the GPTMMIS register.
3	RTCCINT	W1C	0	GPTM RTC Interrupt Clear Writing a 1 to this bit clears the RTCRIS bit in the GPTMRIS register and the RTCMIS bit in the GPTMMIS register.
2	CAECINT	W1C	0	GPTM Capture A Event Interrupt Clear Writing a 1 to this bit clears the CAERIS bit in the GPTMRIS register and the CAEMIS bit in the GPTMMIS register.

Bit/Field	Name	Type	Reset	Description
1	CAMCINT	W1C	0	GPTM Capture A Match Interrupt Clear Writing a 1 to this bit clears the CAMRIS bit in the GPTMRIS register and the CAMMIS bit in the GPTMMIS register.
0	TATOCINT	W1C	0	GPTM Timer A Time-Out Raw Interrupt Writing a 1 to this bit clears the TATORIS bit in the GPTMRIS register and the TATOMIS bit in the GPTMMIS register.

Register 9: GPTM Timer A Interval Load (GPTMTAILR), offset 0x028

When the timer is counting down, this register is used to load the starting count value into the timer. When the timer is counting up, this register sets the upper bound for the timeout event.

When a 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 a 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)

Timer 0 base: 0x4003.0000

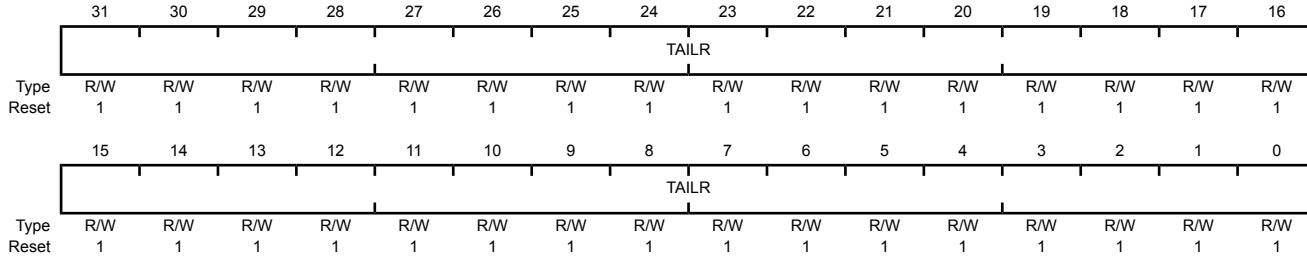
Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 base: 0x4003.3000

Offset 0x028

Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	TAILR	R/W	0xFFFF.FFFF	GPTM Timer A Interval Load Register Writing this field loads the counter for Timer A. A read returns the current value of GPTMTAILR .

Register 10: GPTM Timer B Interval Load (GPTMTBILR), offset 0x02C

When the timer is counting down, this register is used to load the starting count value into the timer.
When the timer is counting up, this register sets the upper bound for the timeout event.

When a GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the **GPTMTAILR** register. Reads from this register return the current value of Timer B and writes are ignored. In a 16-bit mode, bits 15:0 are used for the load value. Bits 31:16 are reserved in both cases.

GPTM Timer B Interval Load (GPTMTBILR)

Timer 0 base: 0x4003.0000

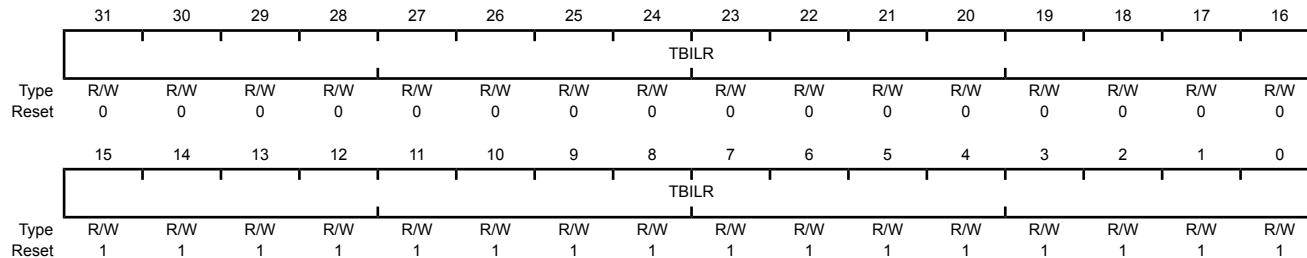
Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 base: 0x4003.3000

Offset 0x02C

Type R/W, reset 0x0000.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	TBILR	R/W	0x0000.FFFF	GPTM Timer B Interval Load Register Writing this field loads the counter for Timer B. A read returns the current value of GPTMTBILR . When a GPTM is 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 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.

In PWM mode, this value along with **GPTMTAILR**, determines the duty cycle of the output PWM signal.

When a GPTM is configured to one of the 32-bit modes, **GPTMTAMATCHR** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM Timer B Match**

(GPTMTBMATCHR) register). In a 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of **GPTMTBMATCHR**.

GPTM Timer A Match (GPTMTAMATCHR)

Timer 0 base: 0x4003.0000

Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 base: 0x4003.3000

Offset 0x030

Type R/W, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMR																
Type	R/W															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
TAMR																
Type	R/W															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:0	TAMR	R/W	0xFFFF.FFFF	GPTM Timer A Match Register This value is compared to the GPTMTAR register to determine match events.

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 Edge-Count mode, this register along with **GPTMTBILR**, determines how many edge events are counted. The total number of edge events counted is equal to the value in **GPTMTBILR** minus this value.

In PWM mode, this value along with **GPTMTBILR**, determines the duty cycle of the output PWM signal.

When a GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the **GPTMTAMATCHR** register. Reads from this register return the current match value of Timer B and writes are ignored. In a 16-bit mode, bits 15:0 are used for the match value. Bits 31:16 are reserved in both cases.

GPTM Timer B Match (GPTMTBMATCHR)

Timer 0 base: 0x4003.0000

Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 base: 0x4003.3000

Offset 0x034

Type R/W, reset 0x0000.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TBMR															
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
	TBMR															
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	TBMR	R/W	0x0000.FFFF	GPTM Timer B Match Register This value is compared to the GPTMTBR register to determine match events.

Register 13: GPTM Timer A Prescale (GPTMTAPR), offset 0x038

This register allows software to extend the range of the 16-bit timers in periodic and one-shot modes. In Edge-Count mode, this register is the MSB of the 24-bit count value.

GPTM Timer A Prescale (GPTMTAPR)

Timer 0 base: 0x4003.0000

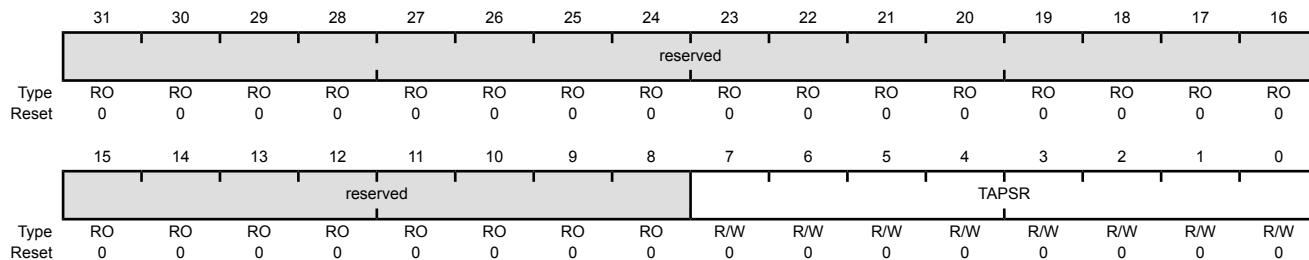
Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 base: 0x4003.3000

Offset 0x038

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	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 10-5 on page 485 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 in periodic and one-shot modes. In Edge-Count mode, this register is the MSB of the 24-bit count value.

GPTM Timer B Prescale (GPTMTBPR)

Timer 0 base: 0x4003.0000

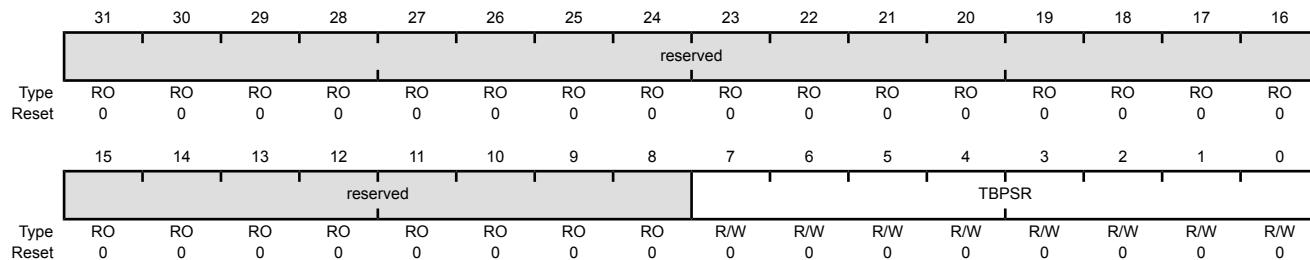
Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 base: 0x4003.3000

Offset 0x03C

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	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 10-5 on page 485 for more details and an example.

Register 15: GPTM TimerA Prescale Match (GPTMTAPMR), offset 0x040

This register effectively extends the range of **GPTMTAMATCHR** to 24 bits when operating in 16-bit one-shot or periodic mode.

GPTM TimerA Prescale Match (GPTMTAPMR)

Timer 0 base: 0x4003.0000

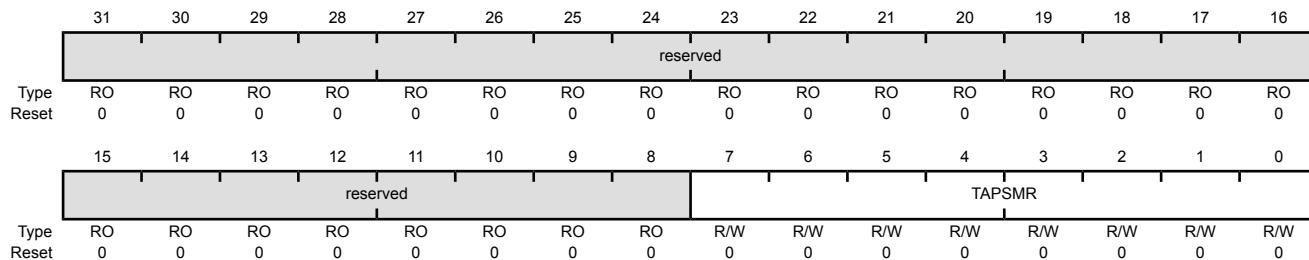
Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 base: 0x4003.3000

Offset 0x040

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TAPSMR	R/W	0x00	GPTM TimerA Prescale Match This value is used alongside GPTMTAMATCHR to detect timer match events while using a prescaler.

Register 16: GPTM TimerB Prescale Match (GPTMTBPMR), offset 0x044

This register effectively extends the range of **GPTMTBMR** to 24 bits when operating in 16-bit one-shot or periodic mode.

GPTM TimerB Prescale Match (GPTMTBPMR)

Timer 0 base: 0x4003.0000

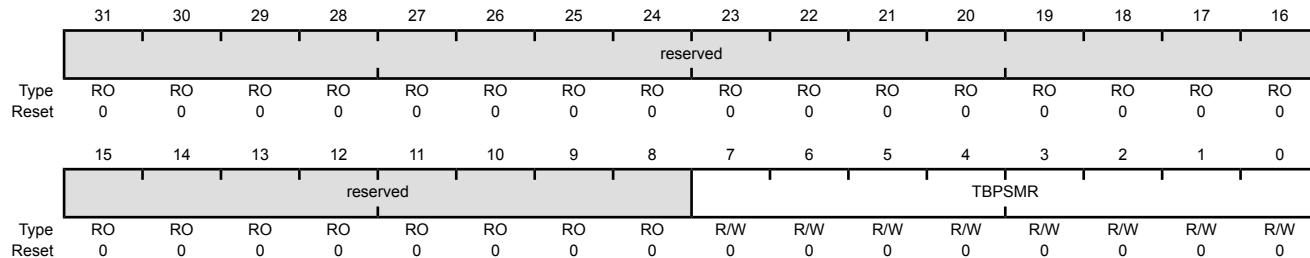
Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 base: 0x4003.3000

Offset 0x044

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TBPSMR	R/W	0x00	GPTM TimerB Prescale Match This value is used alongside GPTMTBMR to detect timer match events while using a prescaler.

Register 17: 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 and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place. Also in Input Edge-Count mode, bits 23:16 contain the upper 8 bits of the count.

When a GPTM is configured to one of the 32-bit modes, **GPTMTAR** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM Timer B (GPTMTBR)** register). In the 16-bit Input Edge Count, Input Edge Time, and PWM modes, bits 15:0 contain the value of the counter and bits 23:16 contain the value of the prescaler, which is the upper 8 bits of the count. Bits 31:24 always read as 0. To read the value of the prescaler in 16-bit One-Shot and Periodic modes, read bits [23:16] in the **GPTMTAV** register.

GPTM Timer A (GPTMTAR)

Timer 0 base: 0x4003.0000

Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 base: 0x4003.3000

Offset 0x048

Type RO, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TAR															
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
	TAR															
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	TAR	RO	0xFFFF.FFFF	GPTM Timer A Register A read returns the current value of the GPTM Timer A Count Register , in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.

Register 18: 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 and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place. Also in Input Edge-Count mode, bits 23:16 contain the upper 8 bits of the count.

When a GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the **GPTMTAR** register. Reads from this register return the current value of Timer B. In a 16-bit mode, bits 15:0 contain the value of the counter and bits 23:16 contain the value of the prescaler in Input Edge Count, Input Edge Time, and PWM modes, which is the upper 8 bits of the count. Bits 31:24 are reserved in both cases.

GPTM Timer B (GPTMTBR)

Timer 0 base: 0x4003.0000

Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 base: 0x4003.3000

Offset 0x04C

Type RO, reset 0x0000.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TBR															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TBR															
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	TBR	RO	0x0000.FFFF	GPTM Timer B Register A read returns the current value of the GPTM Timer B Count Register , in all cases except for Input Edge Count and Time modes. In the Input Edge Count mode, this register contains the number of edges that have occurred. In the Input Edge Time mode, this register contains the time at which the last edge event took place.

Register 19: GPTM Timer A Value (GPTMTAV), offset 0x050

When read, 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. When written, the value written into this register is loaded into the **GPTMTAR** register on the next clock cycle. In Input Edge-Count mode, bits 23:16 contain the upper 8 bits of the count.

When a GPTM is configured to one of the 32-bit modes, **GPTMTAV** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM Timer B Value (GPTMTBV)** register). In a 16-bit mode, bits 15:0 contain the value of the counter and bits 23:16 contain the current, free-running value of the prescaler, which is the upper 8 bits of the count. Bits 31:24 always read as 0.

Note: The **GPTMTAV** register cannot be written in Edge-Count mode.

GPTM Timer A Value (GPTMTAV)

Timer 0 base: 0x4003.0000

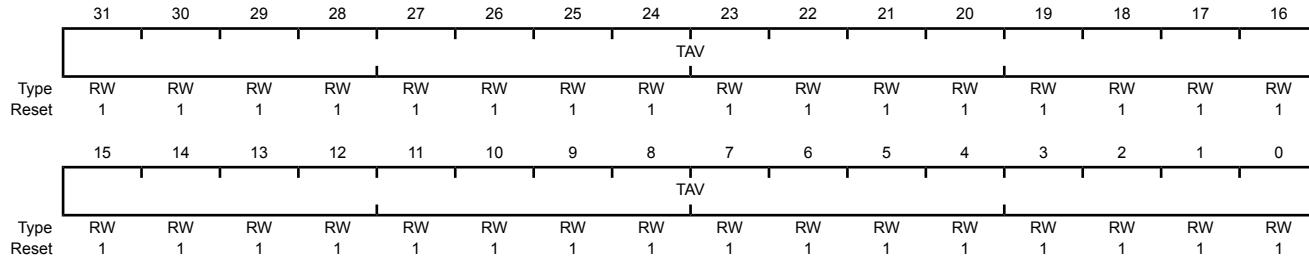
Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 base: 0x4003.3000

Offset 0x050

Type RW, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	TAV	RW	0xFFFF.FFFF	GPTM Timer A Value A read returns the current, free-running value of Timer A in all modes. When written, the value written into this register is loaded into the GPTMTAR register on the next clock cycle.

Register 20: GPTM Timer B Value (GPTMTBV), offset 0x054

When read, 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. When written, the value written into this register is loaded into the **GPTMTBR** register on the next clock cycle. In Input Edge-Count mode, bits 23:16 contain the upper 8 bits of the count.

When a GPTM is configured to one of the 32-bit modes, the contents of bits 15:0 in this register are loaded into the upper 16 bits of the **GPTMTAV** register. Reads from this register return the current free-running value of Timer B. In a 16-bit mode, bits 15:0 contain the current, free-running value of the counter and bits 23:16 contain the current, free-running value of the prescaler, which is the upper 8 bits of the count. Bits 31:24 are reserved in both cases.

GPTM Timer B Value (GPTMTBV)

Timer 0 base: 0x4003.0000

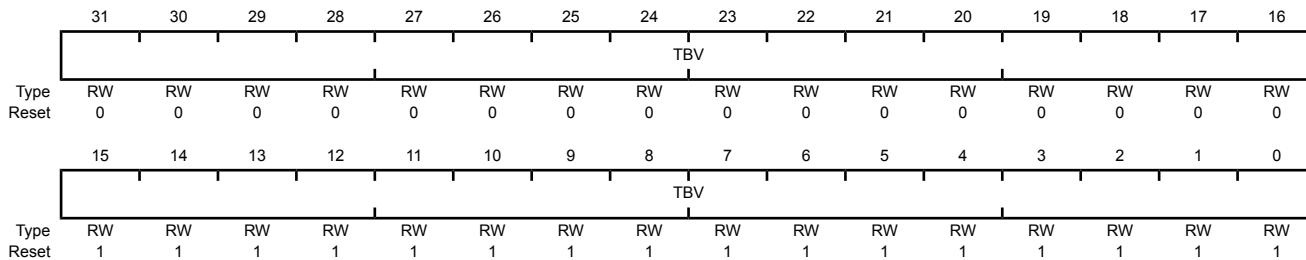
Timer 1 base: 0x4003.1000

Timer 2 base: 0x4003.2000

Timer 3 base: 0x4003.3000

Offset 0x054

Type RW, reset 0x0000.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	TBV	RW	0x0000.FFFF	GPTM Timer B Value A read returns the current, free-running value of Timer A in all modes. When written, the value written into this register is loaded into the GPTMTAR register on the next clock cycle.

11 Watchdog Timers

A watchdog timer can generate an interrupt or a reset when a time-out value is reached. The watchdog timer is used to regain control when a system has failed due to a software error or due to the failure of an external device to respond in the expected way. The LM3S9L97 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.

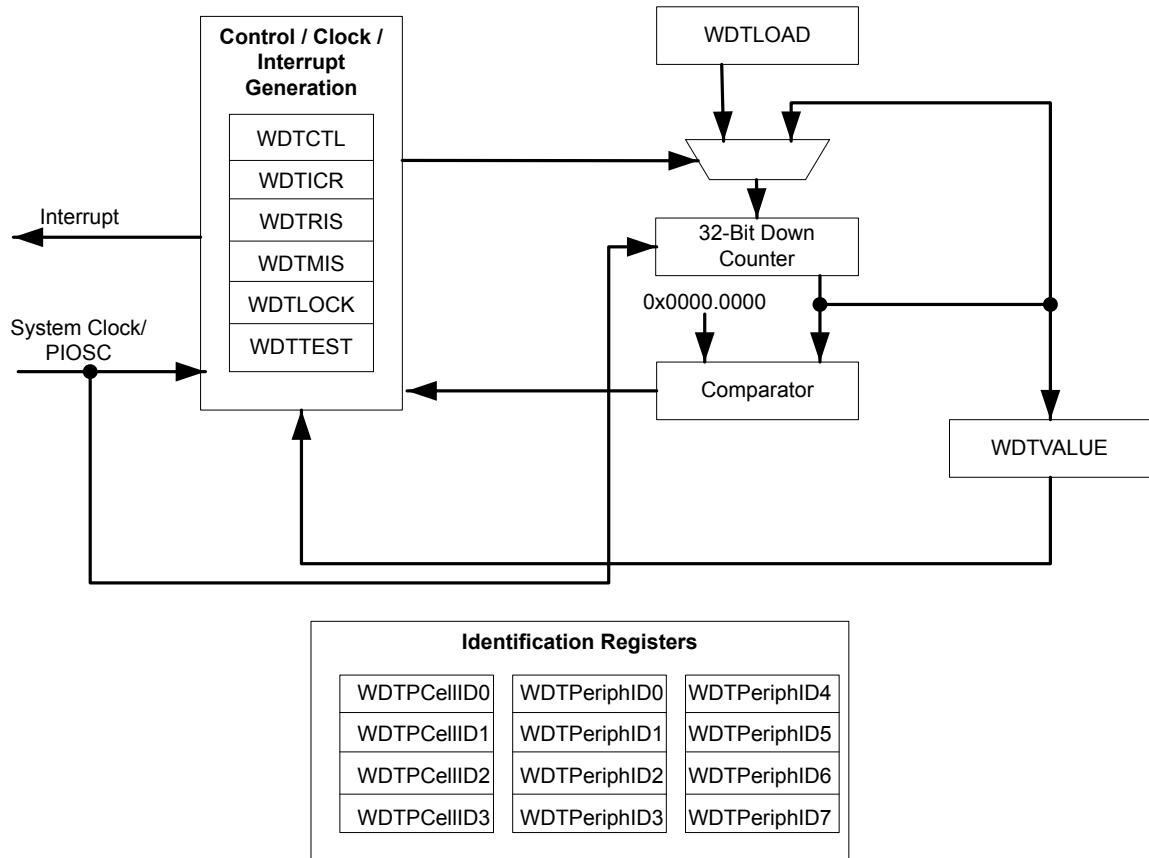
The Stellaris® LM3S9L97 controller has two Watchdog Timer modules with the following features:

- 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 microcontroller 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.

11.1 Block Diagram

Figure 11-1. WDT Module Block Diagram



11.2 Functional Description

The Watchdog Timer module generates the first time-out signal when the 32-bit counter reaches the zero state after being enabled; enabling the counter also enables the watchdog timer interrupt. After the first time-out event, the 32-bit counter is re-loaded with the value of the **Watchdog Timer Load (WDTLOAD)** register, and the timer resumes counting down from that value. Once the Watchdog Timer has been configured, the **Watchdog Timer Lock (WDTLOCK)** register is written, which prevents the timer configuration from being inadvertently altered by software.

If the timer counts down to its zero state again before the first time-out interrupt is cleared, and the reset signal has been enabled 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.

11.2.1 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.

11.3 Initialization and Configuration

To use the WDT, its peripheral clock must be enabled by setting the **WDT** bit in the **RCGC0** register, see page 268.

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.

11.4 Register Map

Table 11-1 on page 528 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 268).

Table 11-1. Watchdog Timers Register Map

Offset	Name	Type	Reset	Description	See page
0x000	WDTLOAD	R/W	0xFFFF.FFFF	Watchdog Load	529
0x004	WDTVALUE	RO	0xFFFF.FFFF	Watchdog Value	530
0x008	WDTCTL	R/W	0x0000.0000 (WDT0) 0x8000.0000 (WDT1)	Watchdog Control	531
0x00C	WDTICR	WO	-	Watchdog Interrupt Clear	533
0x010	WDTRIS	RO	0x0000.0000	Watchdog Raw Interrupt Status	534
0x014	WDTMIS	RO	0x0000.0000	Watchdog Masked Interrupt Status	535
0x418	WDTTEST	R/W	0x0000.0000	Watchdog Test	536
0xC00	WDTLOCK	R/W	0x0000.0000	Watchdog Lock	537
0xFD0	WDTPeriphID4	RO	0x0000.0000	Watchdog Peripheral Identification 4	538
0xFD4	WDTPeriphID5	RO	0x0000.0000	Watchdog Peripheral Identification 5	539
0xFD8	WDTPeriphID6	RO	0x0000.0000	Watchdog Peripheral Identification 6	540
0xFDC	WDTPeriphID7	RO	0x0000.0000	Watchdog Peripheral Identification 7	541
0xFE0	WDTPeriphID0	RO	0x0000.0005	Watchdog Peripheral Identification 0	542
0xFE4	WDTPeriphID1	RO	0x0000.0018	Watchdog Peripheral Identification 1	543
0xFE8	WDTPeriphID2	RO	0x0000.0018	Watchdog Peripheral Identification 2	544
0xFEC	WDTPeriphID3	RO	0x0000.0001	Watchdog Peripheral Identification 3	545
0xFF0	WDTPCellID0	RO	0x0000.000D	Watchdog PrimeCell Identification 0	546
0xFF4	WDTPCellID1	RO	0x0000.00F0	Watchdog PrimeCell Identification 1	547
0xFF8	WDTPCellID2	RO	0x0000.0006	Watchdog PrimeCell Identification 2	548
0xFFC	WDTPCellID3	RO	0x0000.00B1	Watchdog PrimeCell Identification 3	549

11.5 Register Descriptions

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

Register 1: Watchdog Load (WDTLOAD), offset 0x000

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

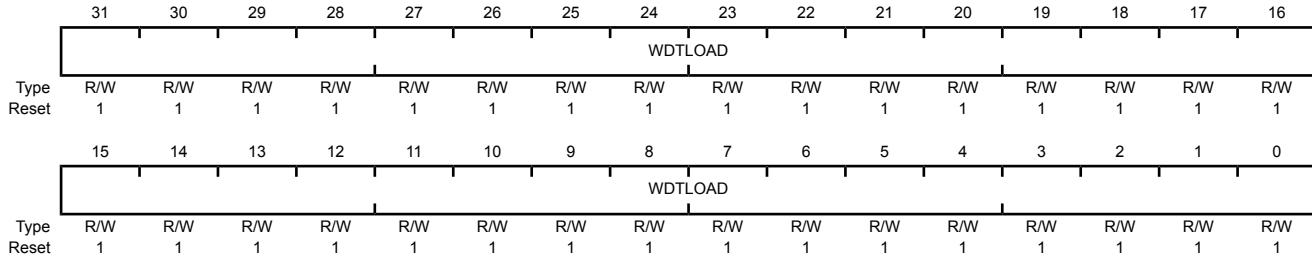
Watchdog Load (WDTLOAD)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0x000

Type R/W, reset 0xFFFF.FFFF



Bit/Field

Name

Type

Reset

Description

31:0 WDTLOAD R/W 0xFFFF.FFFF Watchdog Load Value

Register 2: Watchdog Value (WDTVALUE), offset 0x004

This register contains the current count value of the timer.

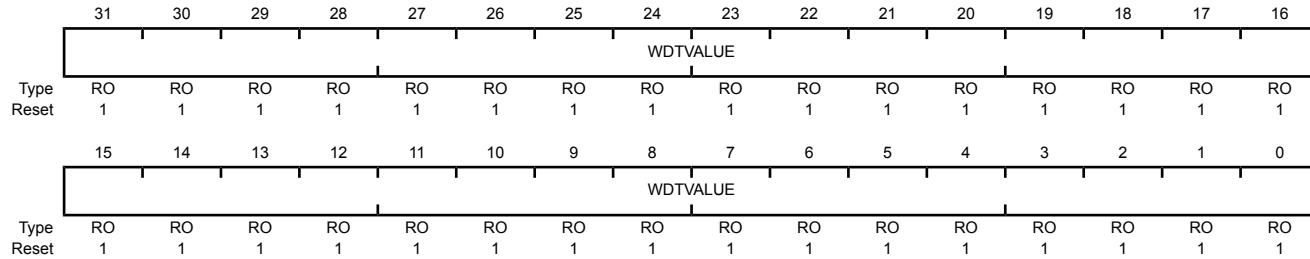
Watchdog Value (WDTVALUE)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0x004

Type RO, reset 0xFFFF.FFFF



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

31:0	WDTVALUE	RO	0xFFFF.FFFF	Watchdog Value Current value of the 32-bit down counter.
------	----------	----	-------------	-------------------------------------------------------------

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

This register is the watchdog control register. The watchdog timer can be configured to generate a reset signal (on second time-out) or an interrupt on time-out.

When the watchdog interrupt has been enabled, all subsequent writes to the control register are ignored. The only mechanism that can re-enable writes is a hardware reset.

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 (WDT0) and 0x8000.0000 (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 A write access is not in progress, 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.

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.

Bit/Field	Name	Type	Reset	Description
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.

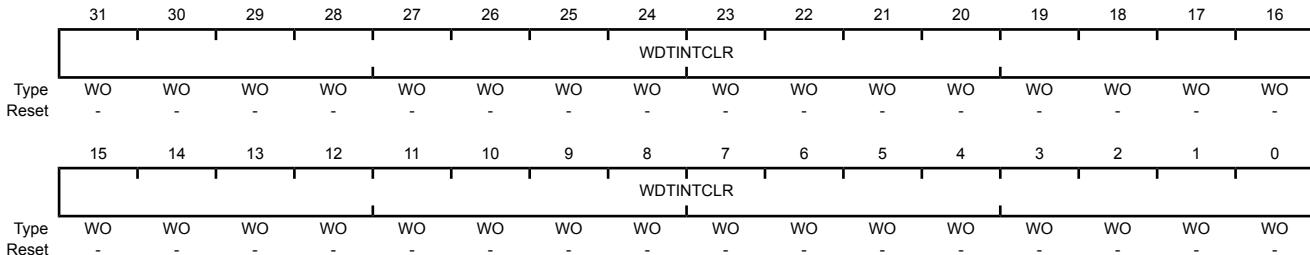
Watchdog Interrupt Clear (WDTICR)

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.

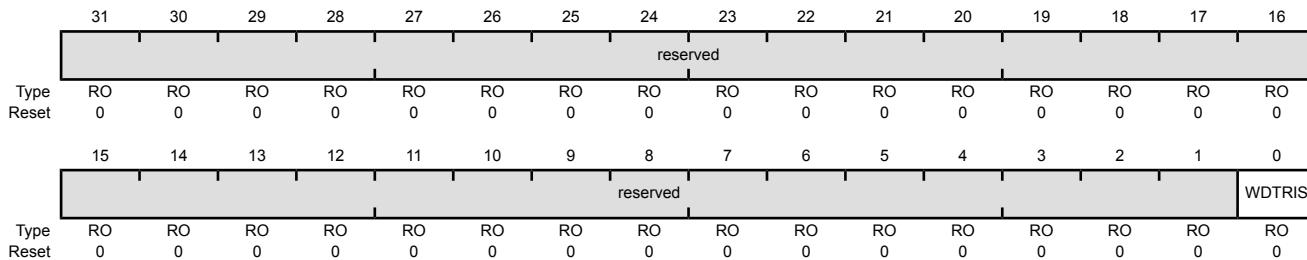
Watchdog Raw Interrupt Status (WDTRIS)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0x010

Type RO, reset 0x0000.0000



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	WDTRIS	RO	0	Watchdog Raw Interrupt Status
		Value	Description	
		1	A watchdog time-out event has occurred.	
		0	The watchdog has not timed out.	

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.

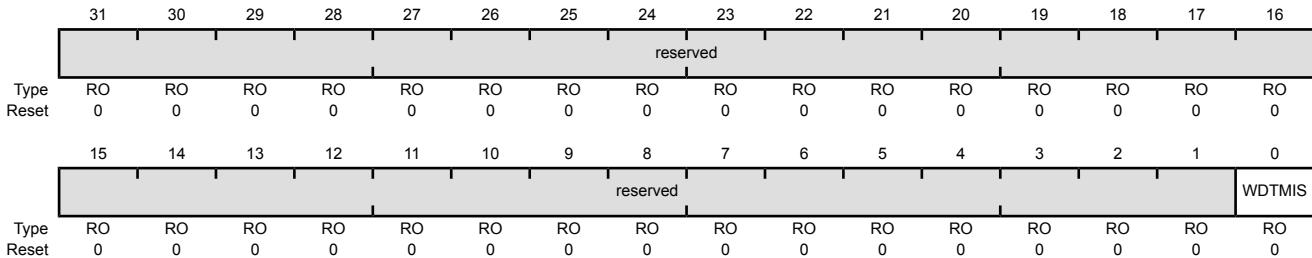
Watchdog Masked Interrupt Status (WDTMIS)

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	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	WDTMIS	RO	0	Watchdog Masked Interrupt Status
		Value	Description	
		1	A watchdog time-out event has been signalled to the interrupt controller.	
		0	The watchdog has not timed out or the watchdog timer interrupt is masked.	

Register 7: Watchdog Test (WDTTEST), offset 0x418

This register provides user-enabled stalling when the microcontroller asserts the CPU halt flag during debug.

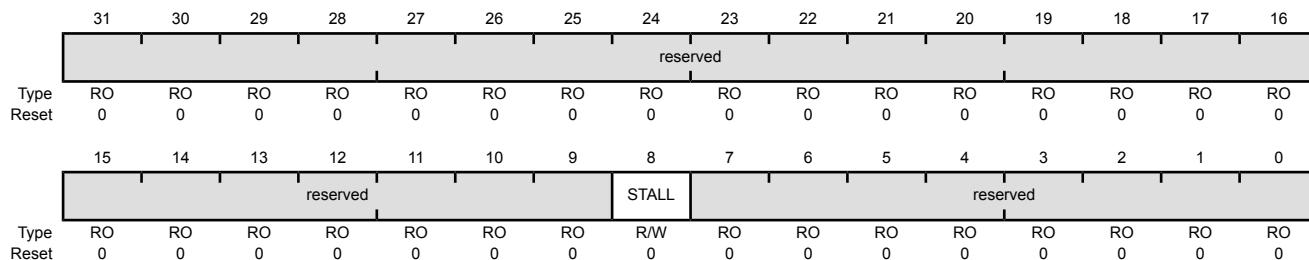
Watchdog Test (WDTTEST)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0x418

Type R/W, reset 0x0000.0000



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	STALL	R/W	0	Watchdog Stall Enable
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.

Value Description

- 1 If the microcontroller is stopped with a debugger, the watchdog timer stops counting. Once the microcontroller is restarted, the watchdog timer resumes counting.
- 0 The watchdog timer continues counting if the microcontroller is stopped with a debugger.

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)).

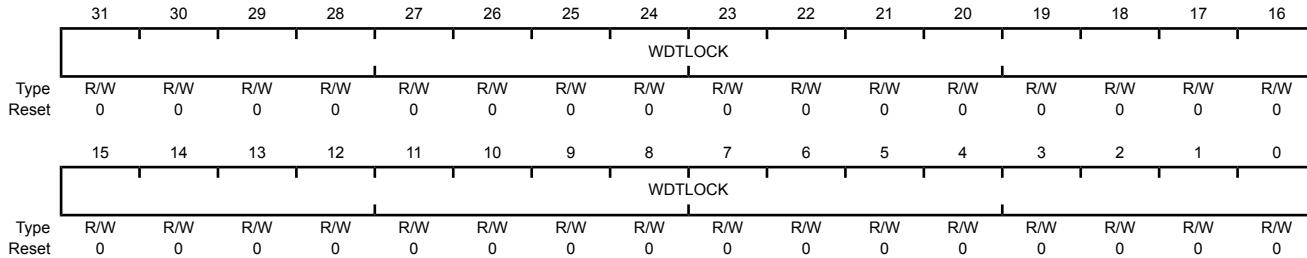
Watchdog Lock (WDTLOCK)

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 0x0000.0000 Watchdog Lock

A write of the value 0x1ACC.E551 unlocks the watchdog registers for write access. A write of any other value re-applies the lock, preventing any register updates.

A read of this register returns the following values:

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

Register 9: Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0

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

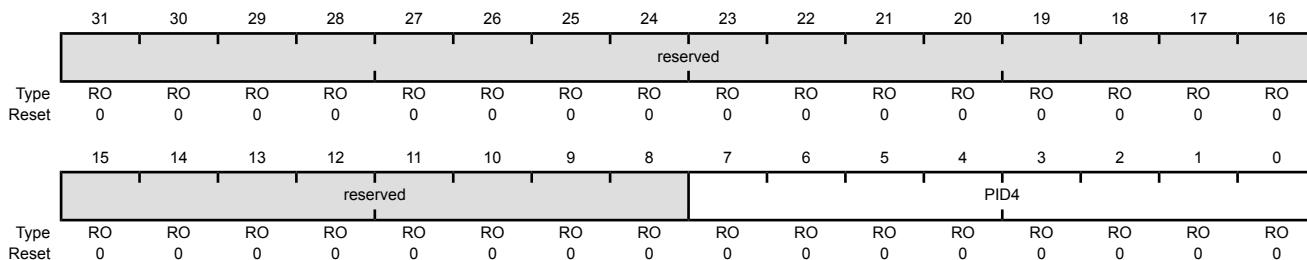
Watchdog Peripheral Identification 4 (WDTPeriphID4)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

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	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.

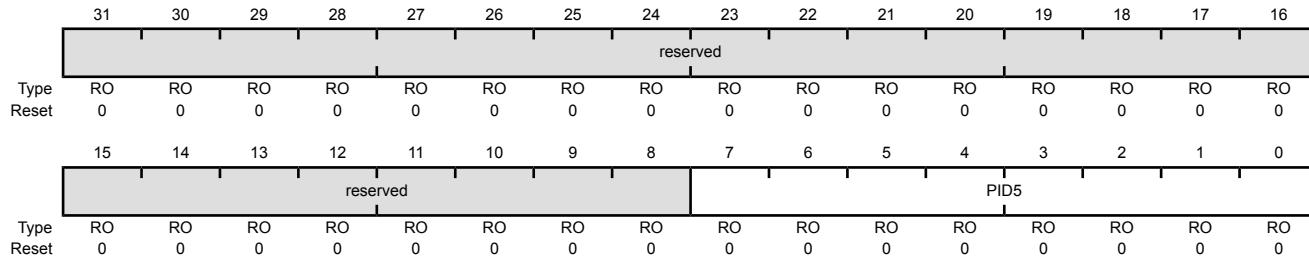
Watchdog Peripheral Identification 5 (WDTPeriphID5)

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	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	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.

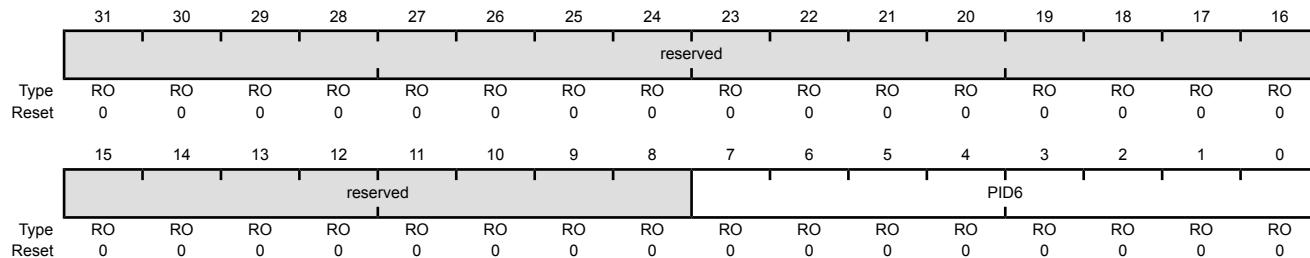
Watchdog Peripheral Identification 6 (WDTPeriphID6)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFD8

Type RO, reset 0x0000.0000



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	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.

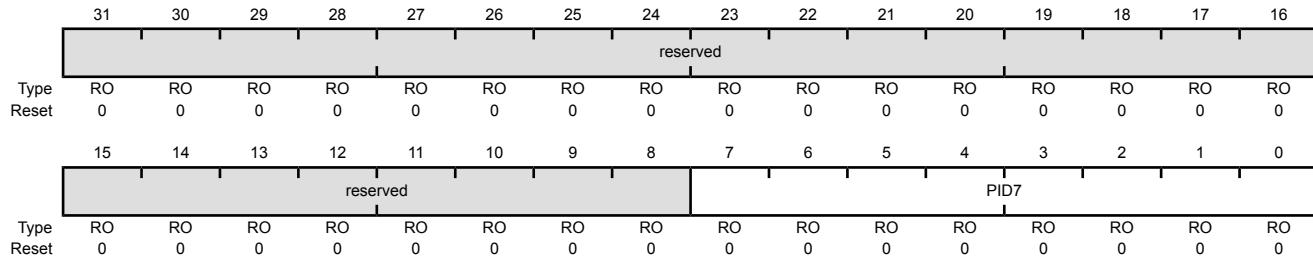
Watchdog Peripheral Identification 7 (WDTPeriphID7)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFDC

Type RO, reset 0x0000.0000



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	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.

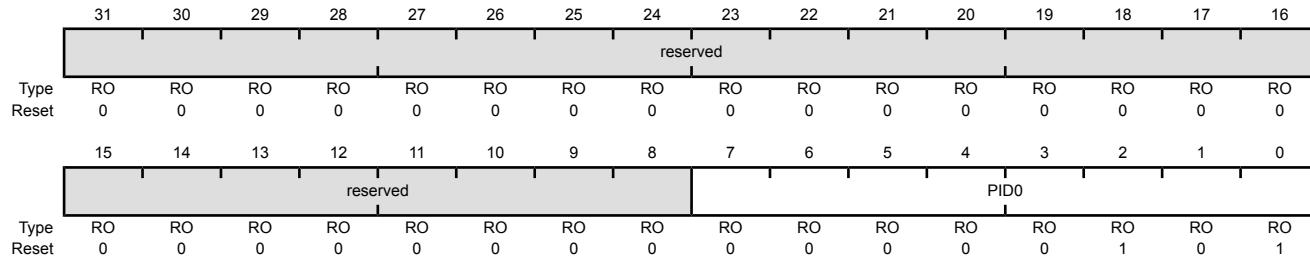
Watchdog Peripheral Identification 0 (WDTPeriphID0)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFE0

Type RO, reset 0x0000.0005



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	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.

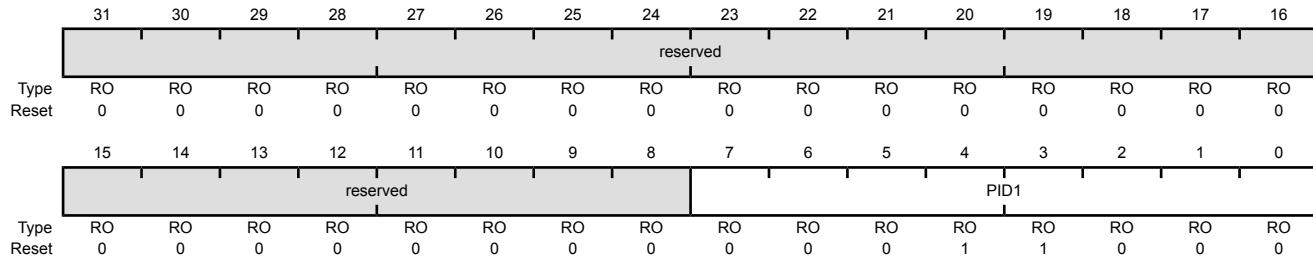
Watchdog Peripheral Identification 1 (WDTPeriphID1)

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	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	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.

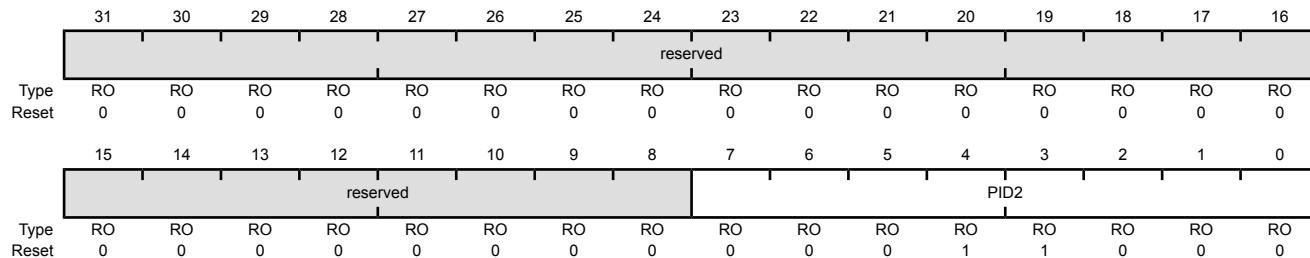
Watchdog Peripheral Identification 2 (WDTPeriphID2)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFE8

Type RO, reset 0x0000.0018



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	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.

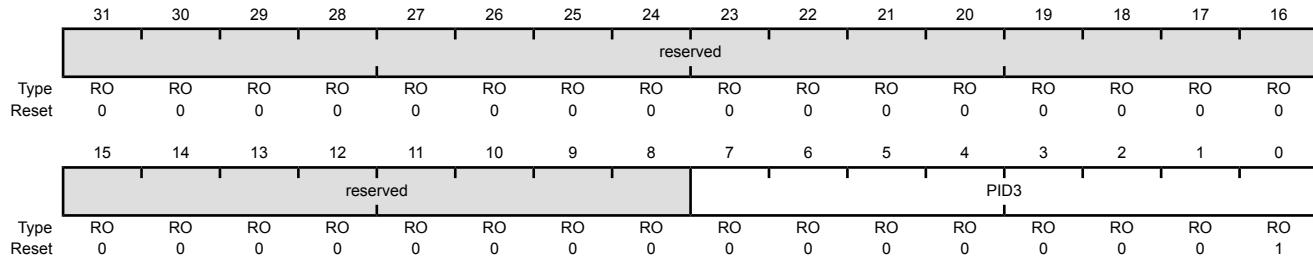
Watchdog Peripheral Identification 3 (WDTPeriphID3)

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	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	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.

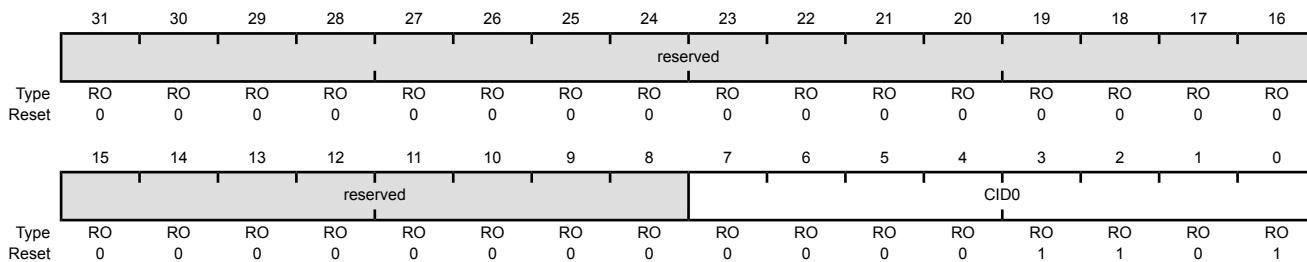
Watchdog PrimeCell Identification 0 (WDTPCellID0)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFF0

Type RO, reset 0x0000.000D



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	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.

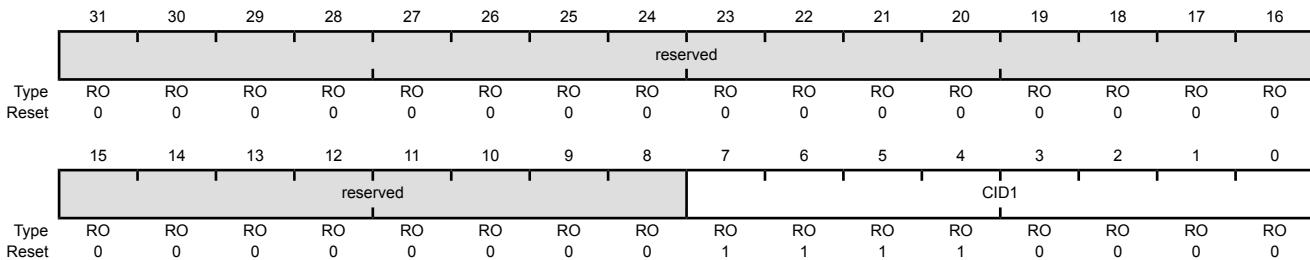
Watchdog PrimeCell Identification 1 (WDTPCellID1)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFF4

Type RO, reset 0x0000.00F0



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	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.

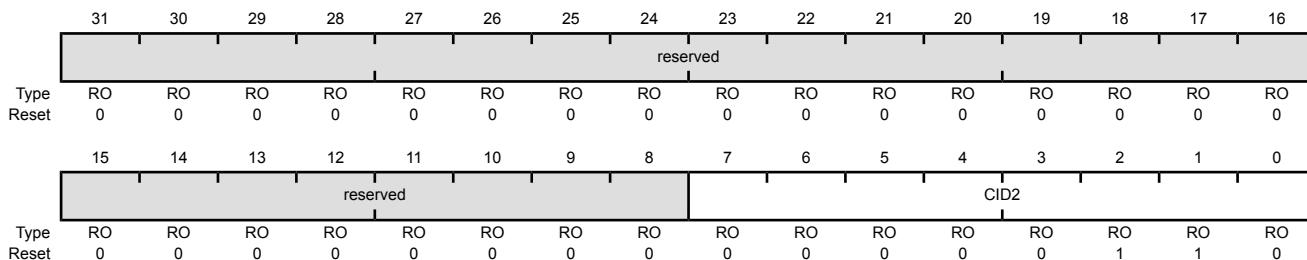
Watchdog PrimeCell Identification 2 (WDTPCellID2)

WDT0 base: 0x4000.0000

WDT1 base: 0x4000.1000

Offset 0xFF8

Type RO, reset 0x0000.0006



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	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.

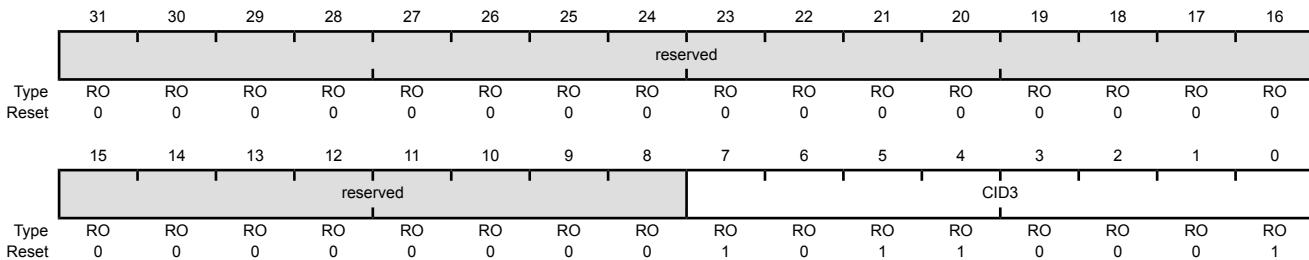
Watchdog PrimeCell Identification 3 (WDTPCellID3)

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	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	Watchdog PrimeCell ID Register [31:24]

12 Analog-to-Digital Converter (ADC)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number. Two identical converter modules are included, which share sixteen input channels.

The Stellaris® ADC module features 10-bit conversion resolution and supports sixteen input channels, plus an internal temperature sensor. Each 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. Each ADC module provides eight digital comparators. Each digital comparator evaluates the ADC conversion value against its two user-defined values to determine the operational range of the signal. The trigger source for ADC0 and ADC1 may be independent or the two ADC modules may operate from the same trigger source and operate on the same or different inputs. A phase shifter can delay the start of sampling by a specified phase angle. When using both ADC modules, it is possible to configure the converters to start the conversions coincidentally or within a relative phase from each other, see “Sample Phase Control” on page 556.

The Stellaris LM3S9L97 microcontroller provides two ADC modules with each having the following features:

- 16 analog input channels
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Maximum sample rate of one million samples/second
- Optional phase shift in sample time programmable from 22.5° to 337.5°
- 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 eight digital comparators
- 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 (μ DMA)
 - Dedicated channel for each sample sequencer
 - ADC module uses burst requests for DMA

12.1 Block Diagram

The Stellaris microcontroller contains two identical Analog-to-Digital Converter modules. 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 12-1 on page 551 shows how the two modules are connected to analog inputs and the system bus.

Figure 12-1. Implementation of Two ADC Blocks

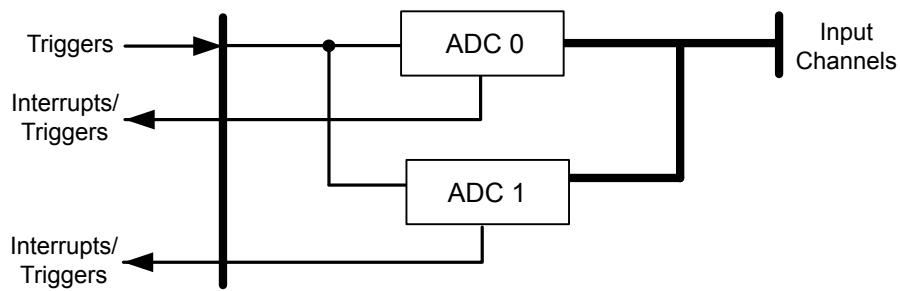
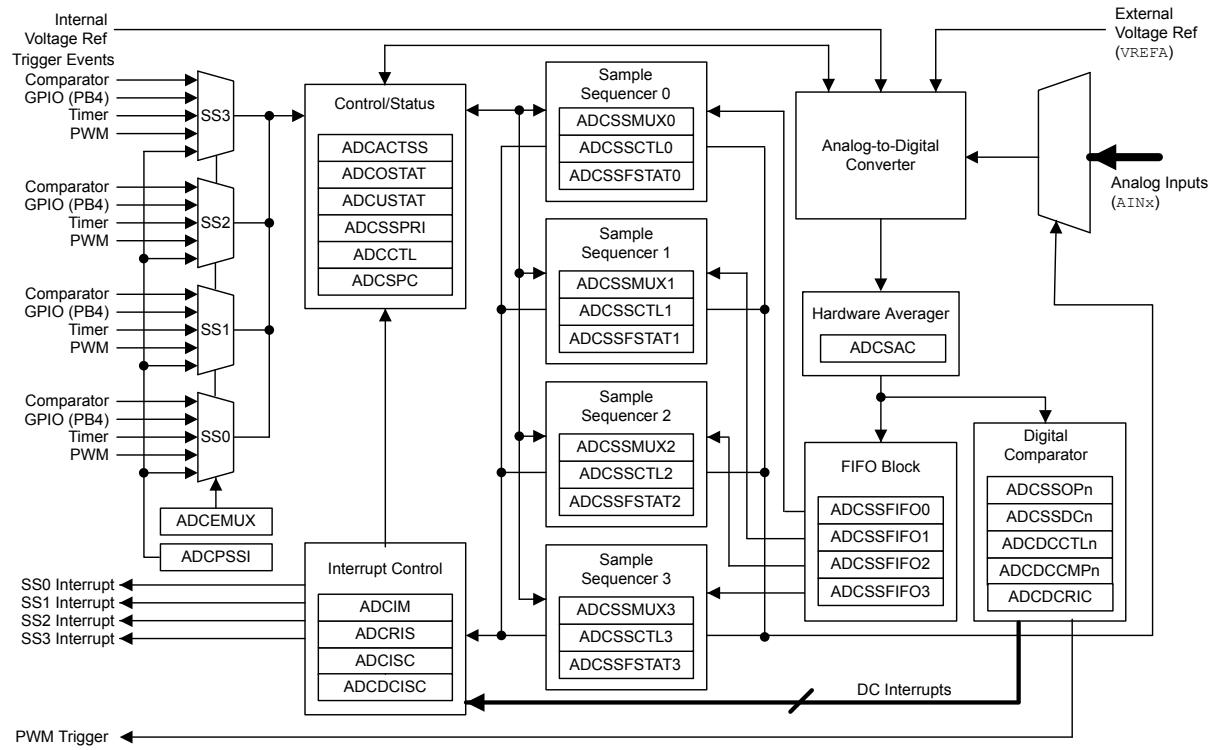


Figure 12-2 on page 552 provides details on the internal configuration of the ADC controls and data registers.

Figure 12-2. ADC Module Block Diagram

12.2 Signal Description

Table 12-1 on page 552 and Table 12-2 on page 553 list the external signals of the ADC module and describe the function of each. The ADC signals are analog functions for some GPIO signals. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the ADC signals. The AIN_x and VREFA analog signals are not 5-V tolerant and go through an isolation circuit before reaching their circuitry. These signals are configured by clearing the corresponding DEN bit in the **GPIO Digital Enable (GPIODEN)** register and setting the corresponding AMSEL bit in the **GPIO Analog Mode Select (GPIOAMSEL)** register. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 423.

Table 12-1. Signals for ADC (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
AIN0	1	PE7	I	Analog	Analog-to-digital converter input 0.
AIN1	2	PE6	I	Analog	Analog-to-digital converter input 1.
AIN2	5	PE5	I	Analog	Analog-to-digital converter input 2.
AIN3	6	PE4	I	Analog	Analog-to-digital converter input 3.
AIN4	100	PD7	I	Analog	Analog-to-digital converter input 4.
AIN5	99	PD6	I	Analog	Analog-to-digital converter input 5.
AIN6	98	PD5	I	Analog	Analog-to-digital converter input 6.
AIN7	97	PD4	I	Analog	Analog-to-digital converter input 7.
AIN8	96	PE3	I	Analog	Analog-to-digital converter input 8.
AIN9	95	PE2	I	Analog	Analog-to-digital converter input 9.

Table 12-1. Signals for ADC (100LQFP) (continued)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
AIN10	92	PB4	I	Analog	Analog-to-digital converter input 10.
AIN11	91	PB5	I	Analog	Analog-to-digital converter input 11.
AIN12	13	PD3	I	Analog	Analog-to-digital converter input 12.
AIN13	12	PD2	I	Analog	Analog-to-digital converter input 13.
AIN14	11	PD1	I	Analog	Analog-to-digital converter input 14.
AIN15	10	PD0	I	Analog	Analog-to-digital converter input 15.
VREFA	90	PB6	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 _n signal is converted to 1023. The VREFA input is limited to the range specified in Table 26-25 on page 1244.

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

Table 12-2. Signals for ADC (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
AIN0	B1	PE7	I	Analog	Analog-to-digital converter input 0.
AIN1	A1	PE6	I	Analog	Analog-to-digital converter input 1.
AIN2	B3	PE5	I	Analog	Analog-to-digital converter input 2.
AIN3	B2	PE4	I	Analog	Analog-to-digital converter input 3.
AIN4	A2	PD7	I	Analog	Analog-to-digital converter input 4.
AIN5	A3	PD6	I	Analog	Analog-to-digital converter input 5.
AIN6	C6	PD5	I	Analog	Analog-to-digital converter input 6.
AIN7	B5	PD4	I	Analog	Analog-to-digital converter input 7.
AIN8	B4	PE3	I	Analog	Analog-to-digital converter input 8.
AIN9	A4	PE2	I	Analog	Analog-to-digital converter input 9.
AIN10	A6	PB4	I	Analog	Analog-to-digital converter input 10.
AIN11	B7	PB5	I	Analog	Analog-to-digital converter input 11.
AIN12	H1	PD3	I	Analog	Analog-to-digital converter input 12.
AIN13	H2	PD2	I	Analog	Analog-to-digital converter input 13.
AIN14	G2	PD1	I	Analog	Analog-to-digital converter input 14.
AIN15	G1	PD0	I	Analog	Analog-to-digital converter input 15.
VREFA	A7	PB6	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 _n signal is converted to 1023. The VREFA input is limited to the range specified in Table 26-25 on page 1244.

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

12.3 Functional Description

The Stellaris ADC collects sample data by using a programmable sequence-based approach instead of the traditional single or double-sampling approaches found on many ADC modules. Each *sample sequence* is a fully programmed series of consecutive (back-to-back) samples, allowing the ADC to collect data from multiple input sources without having to be re-configured or serviced by the 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. In addition, the µDMA can be used to more efficiently move data from the sample sequencers without CPU intervention.

12.3.1 Sample Sequencers

The sampling control and data capture is handled by the sample sequencers. All of the sequencers are identical in implementation except for the number of samples that can be captured and the depth of the FIFO. Table 12-3 on page 554 shows the maximum number of samples that each sequencer can capture and its corresponding FIFO depth. In this implementation, each FIFO entry is a 32-bit word, with the lower 10 bits containing the conversion result.

Table 12-3. Samples and FIFO Depth of Sequencers

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

For a given sample sequence, each sample is defined by bit fields 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** fields select the input pin, while the **ADCSSCTLn** fields 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 590.

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

After a sample sequence completes execution, the result data can be retrieved from the **ADC Sample Sequence Result FIFO (ADCSSFIFOOn)** registers. The FIFOs are simple circular buffers that read a single address to "pop" result data. For software debug purposes, the positions of the FIFO head and tail pointers are visible in the **ADC Sample Sequence FIFO Status (ADCSSFSTATn)** registers along with **FULL** and **EMPTY** status flags. Overflow and underflow conditions are monitored using the **ADCOSTAT** and **ADCUSTAT** registers.

12.3.2 Module Control

Outside of the sample sequencers, the remainder of the control logic is responsible for tasks such as:

- Interrupt generation
- DMA operation
- Sequence prioritization
- Trigger configuration
- Comparator configuration
- External voltage reference
- Sample phase control

Most of the ADC control logic runs at the ADC clock rate of 14-18 MHz. The internal ADC divider is configured for 16-MHz operation automatically by hardware when the system XTAL is selected.

12.3.2.1 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.

12.3.2.2 DMA Operation

The ADC module provides a request signal from each sample sequencer to the associated dedicated channel of the μDMA controller. This configuration allows each sample sequencer to operate independently and transfer data without processor intervention or reconfiguration. The ADC does not support single transfer requests. A burst transfer request is asserted when the interrupt bit for the sample sequence is set (IE bit in the **ADCSSCTLn** register is set).

The arbitration size of the μDMA transfer must be a power of 2, and the associated IE bits in the **ADDSSCTLn** register must be set. For example, if the μDMA 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 μDMA request occurs every time 4 samples have been acquired. No other special steps are needed to enable the ADC module for μDMA operation.

Refer to the “Micro Direct Memory Access (μDMA)” on page 365 for more details about programming the μDMA controller.

12.3.2.3 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.

12.3.2.4 Sampling Events

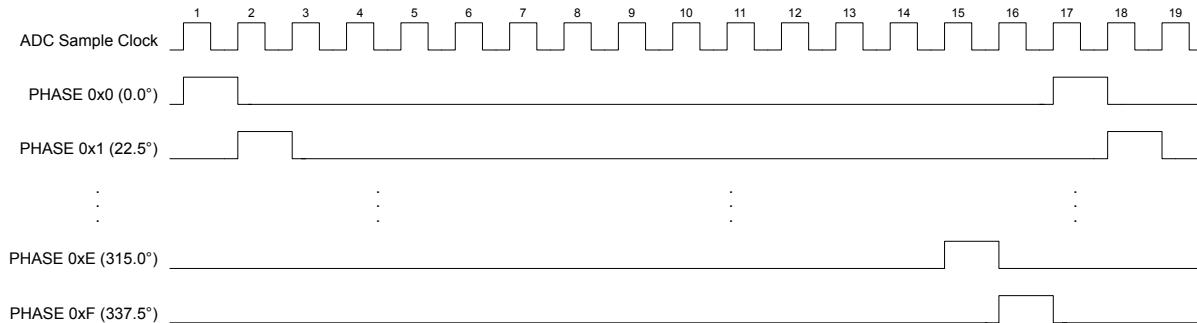
Sample triggering for each sample sequencer is defined in the **ADC Event Multiplexer Select (ADCEMUX)** register. Trigger sources include processor (default), analog comparators, an external signal on GPIO PB4, a GP Timer, PWM2, and continuous sampling. Software can initiate sampling by setting the SS_x bits in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register.

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.

12.3.2.5 Sample Phase Control

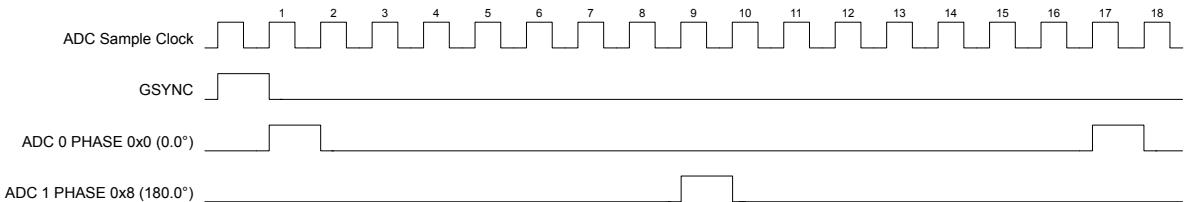
The trigger source for ADC0 and ADC1 may be independent or the two ADC modules may operate from the same trigger source and operate on the same or different inputs. If the converters are running at the same sample rate, they may be configured to start the conversions coincidentally or with one of 15 different discrete phases relative to each other. The sample time can be delayed from the standard sampling time in 22.5° increments up to 337.5° using the **ADC Sample Phase Control (ADCSPC)** register. Figure 12-3 on page 556 shows an example of various phase relationships at a 1 Msps rate.

Figure 12-3. ADC Sample Phases



This feature can be used to double the sampling rate of an input. Both ADC module 0 and ADC module 1 can be programmed to sample the same input. ADC module 0 could sample at the standard position (the PHASE field in the **ADCSPC** register is 0x0). ADC module 1 can be configured to sample at 180 (PHASE = 0x8). The two modules can be synchronized using the GSYNC and SYNCWAIT bits in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register. Software could then combine the results from the two modules to create a sample rate of two million samples/second at 16 MHz as shown in Figure 12-4 on page 556.

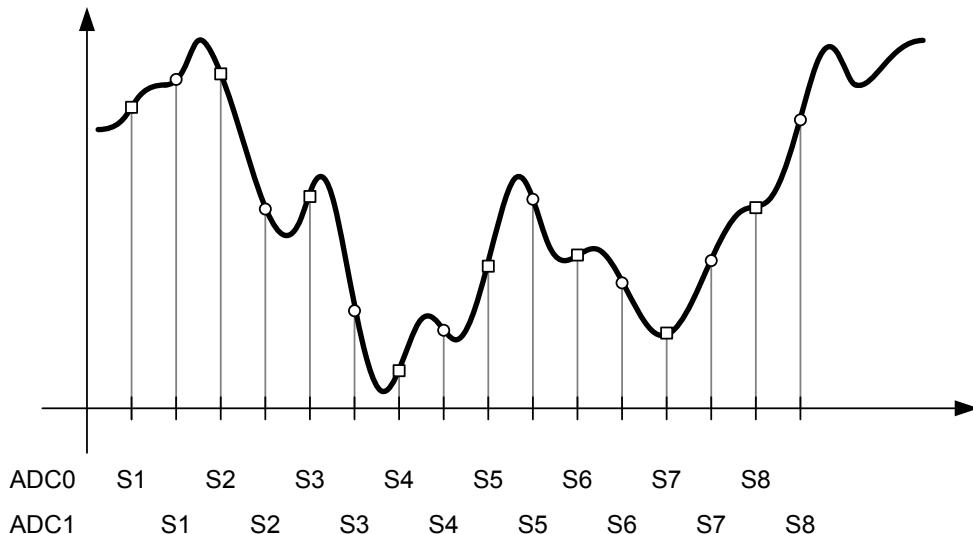
Figure 12-4. Doubling the ADC Sample Rate



Using the **ADCSPC** register, ADC0 and ADC1 may provide a number of interesting applications:

- Coincident sampling of different signals. The sample sequence steps run coincidentally in both converters.
 - ADC Module 0, **ADCSPC** = 0x0, sampling AIN0
 - ADC Module 1, **ADCSPC** = 0x0, sampling AIN1
- Skewed sampling of the same signal. The sample sequence steps are 1/2 of an ADC clock (500 μ s for a 1Ms/s ADC) out of phase with each other. This configuration doubles the conversion bandwidth of a single input when software combines the results as shown in Figure 12-5 on page 557.
 - ADC Module 0, **ADCSPC** = 0x0, sampling AIN0
 - ADC Module 1, **ADCSPC** = 0x8, sampling AIN0

Figure 12-5. Skewed Sampling



12.3.3 Hardware Sample Averaging Circuit

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

By default the averaging circuit is off, and all data from the converter passes through to the sequencer FIFO. The averaging hardware is controlled by the **ADC Sample Averaging Control (ADCSAC)** register (see page 592). A single averaging circuit has been implemented, thus all input channels receive the same amount of averaging whether they are single-ended or differential.

12.3.4 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

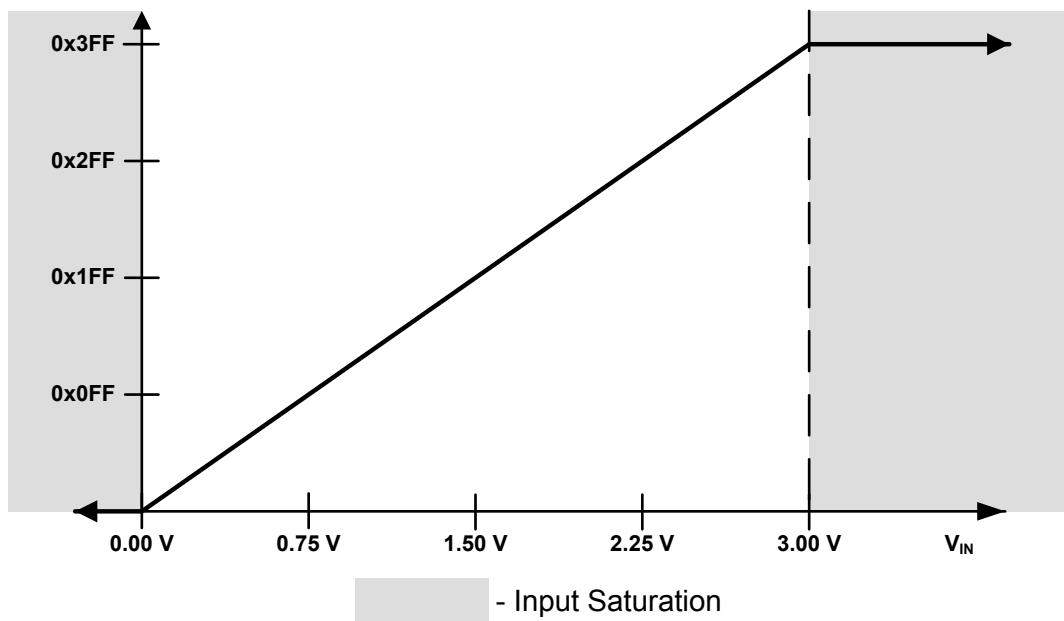
time, resulting in higher conversion speeds for the A/D converter. In addition, built-in sample-and-hold circuitry with offset-calibration circuitry improves conversion accuracy. The ADC must be run from the PLL or a 14- to 18-MHz clock source.

The ADC operates from both the 3.3-V analog and 1.2-V digital power supplies. 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 (ADC)” on page 1243.

12.3.4.1 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. While the analog input pads can handle voltages beyond this range, the ADC conversions saturate in under-voltage and over-voltage cases. Figure 12-6 on page 558 shows the ADC conversion function of the analog inputs.

Figure 12-6. Internal Voltage Conversion Result



12.3.4.2 External Voltage Reference

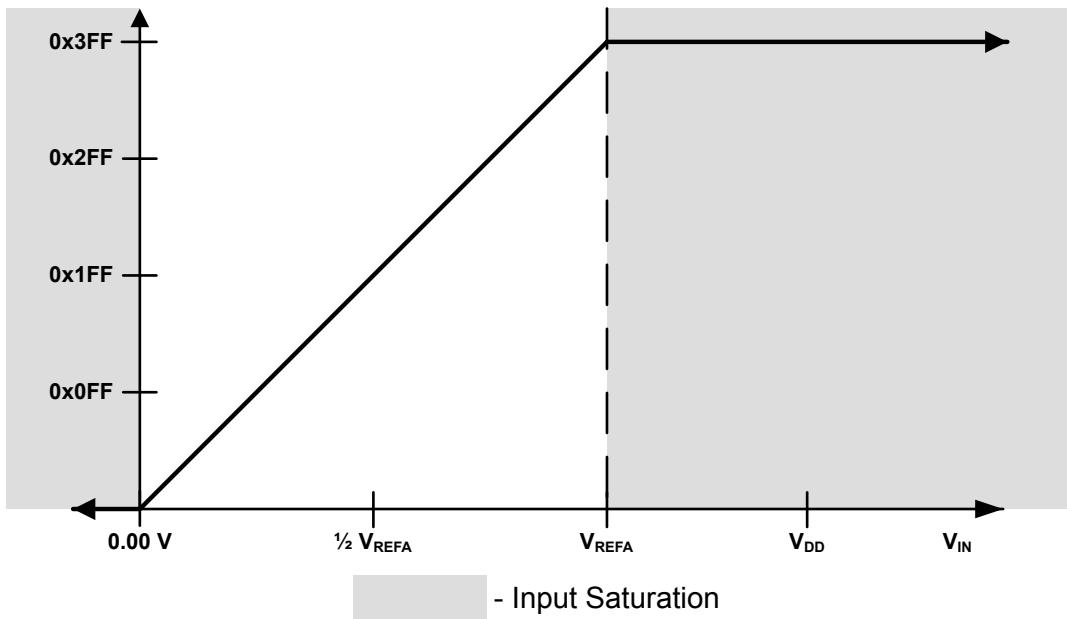
The ADC can use an external voltage reference to produce the conversion value from the selected analog input by setting the VREF bit in the **ADC Control (ADCCTL)** register. The VREF bit specifies whether to use the internal or external reference. While the range of the conversion value remains the same (0x000 to 0x3FF), the analog voltage associated with the 0x3FF value corresponds to the value of the voltage when using the 3.0-V setting and three times the voltage when using the 1.0-V setting, resulting in a smaller voltage resolution per ADC code. Ground is always used as the reference level for the minimum conversion value. Analog input voltages above the external voltage reference saturate to 0x3FF while those below 0.0 V continue to saturate at 0x000. The V_{REFA}

specification defines the useful range for the external voltage reference, see Table 26-25 on page 1244. Care must be taken to supply a reference voltage of acceptable quality.

Figure 12-7 on page 559 shows the ADC conversion function of the analog inputs when using an external voltage reference.

The external voltage reference can be more accurate than the internal reference by using a high-precision source or trimming the source.

Figure 12-7. External Voltage Conversion Result



12.3.5 Differential Sampling

In addition to traditional single-ended sampling, the ADC module supports differential sampling of two analog input channels. To enable differential sampling, software must set the **Dn** bit in the **ADCSSCTL0n** register in a step's configuration nibble.

When a sequence step is configured for differential sampling, 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 12-4 on page 559). The ADC does not support other differential pairings such as analog input 0 with analog input 3.

Table 12-4. 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

Table 12-4. Differential Sampling Pairs (*continued*)

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:

$$\Delta V \text{ (differential voltage)} = V_{\text{IN_EVEN}} \text{ (even channel)} - V_{\text{IN_ODD}} \text{ (odd channel)}, \text{ therefore:}$$

- If $\Delta V = 0$, then the conversion result = 0x1FF
- If $\Delta V > 0$, then the conversion result > 0x1FF (range is 0x1FF–0x3FF)
- If $\Delta V < 0$, then the conversion result < 0x1FF (range is 0–0x1FF)

The differential pairs assign polarities to the analog inputs: the even-numbered input is always positive, and the odd-numbered input is always negative. In order for a valid conversion result to appear, the negative input must be in the range of ± 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 12-8 on page 561 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 12-9 on page 561 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 because the input voltage is less than 0 V. Figure 12-10 on page 562 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.

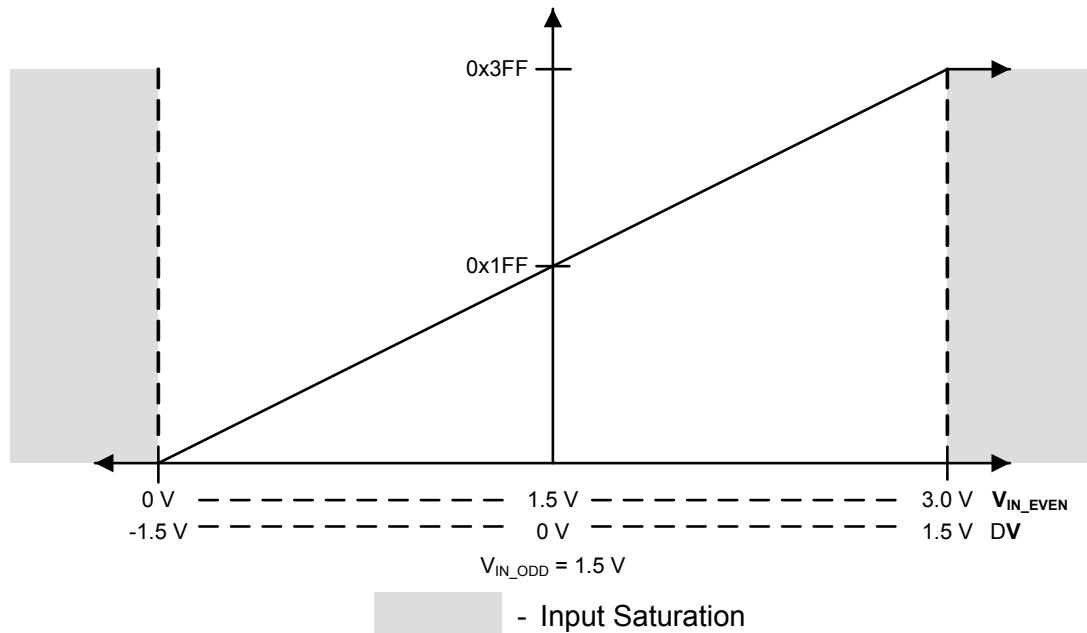
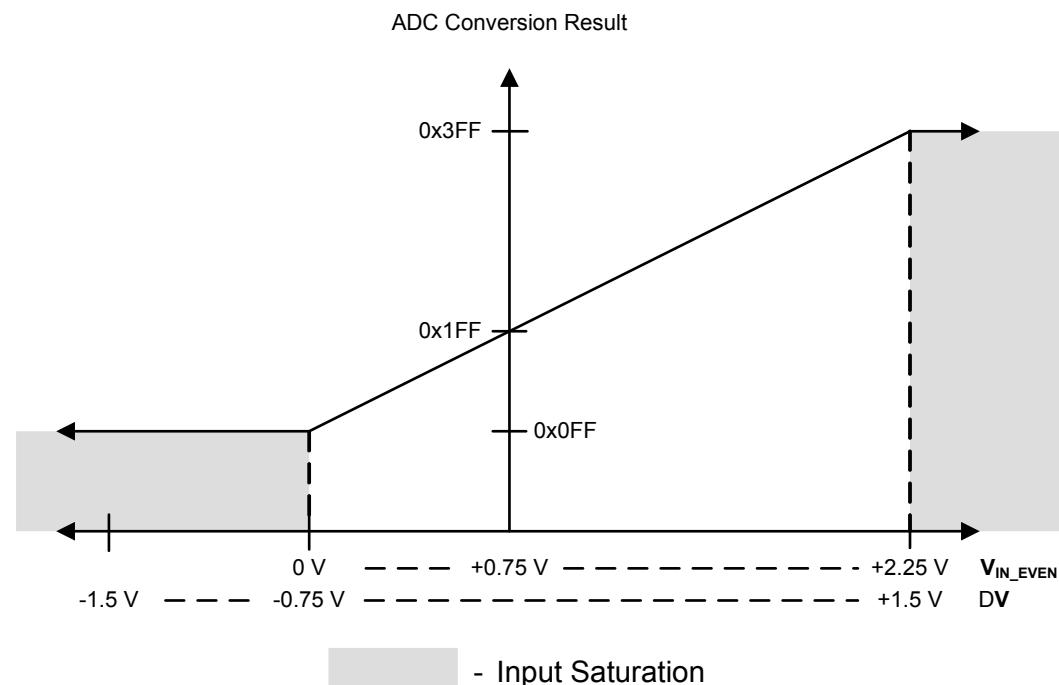
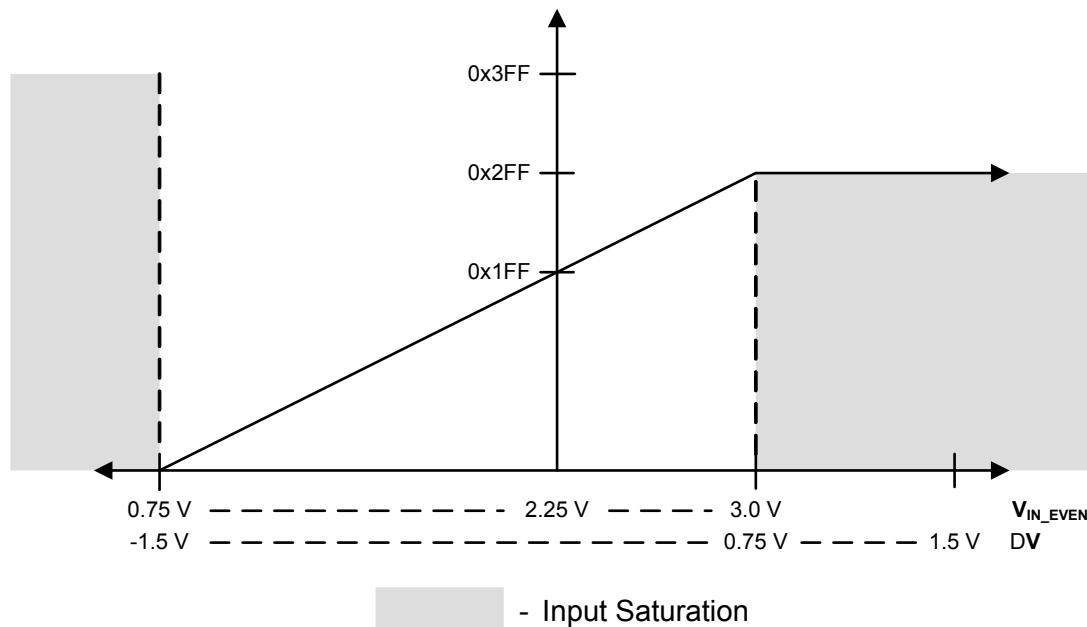
Figure 12-8. Differential Sampling Range, $V_{IN_ODD} = 1.5\text{ V}$ **Figure 12-9. Differential Sampling Range, $V_{IN_ODD} = 0.75\text{ V}$** 

Figure 12-10. Differential Sampling Range, $V_{IN_ODD} = 2.25$ V

12.3.6 Internal Temperature Sensor

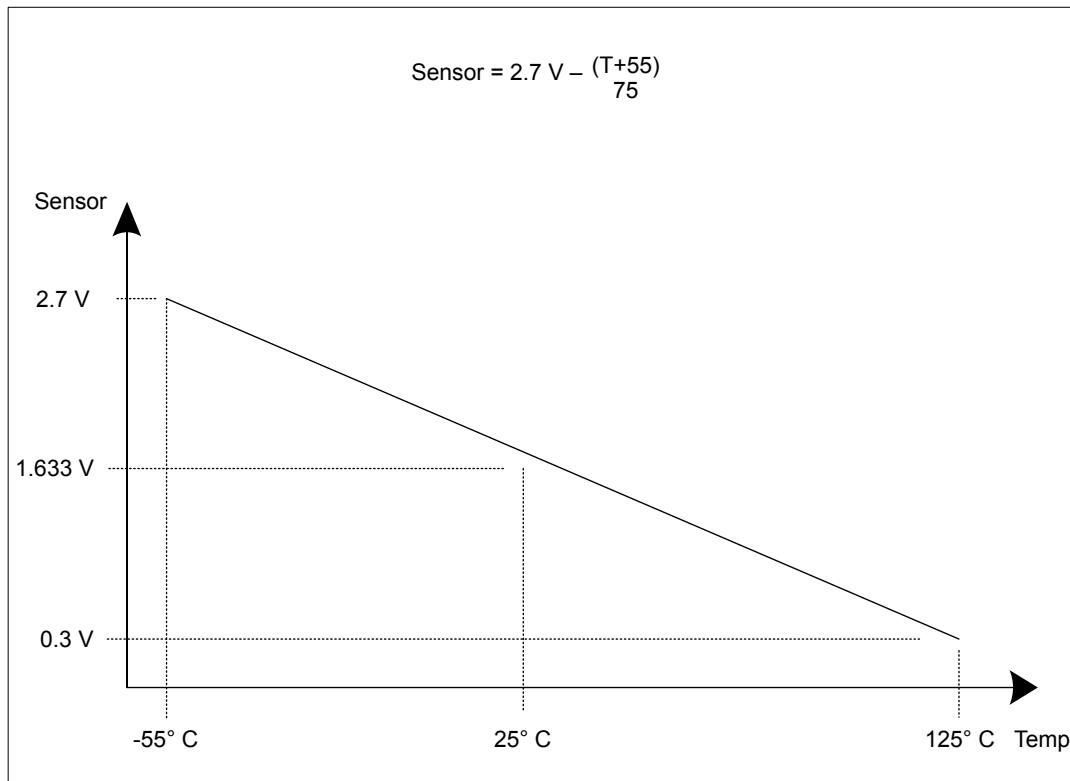
The temperature sensor serves two primary purposes: 1) to notify the system that internal temperature is too high or low for reliable operation and 2) to provide temperature measurements for calibration of the Hibernate module RTC trim value.

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. In addition, the temperature sensor has a second power-down input in the 3.3 V domain which provides control by the Hibernation module.

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 12-11 on page 563.

Figure 12-11. Internal Temperature Sensor Characteristic

The temperature reading from the temperature sensor can also be given as a function of the ADC value. The following formula calculates temperature (in °C) based on the ADC reading:

$$\text{Temperature} = 147.5 - ((225 \times \text{ADC}) / 1023)$$

12.3.7 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, each module provides eight digital comparators. Conversions from the ADC that are sent to the digital comparators are compared against the user programmable limits in the **ADC Digital Comparator Range (ADCDCCMPn)** 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.

12.3.7.1 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_nD COP** 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 (ADCDCCTL n)** 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 **ADCDCCTL n** 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

12.3.7.2 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 **ADCDCCTL n** 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.

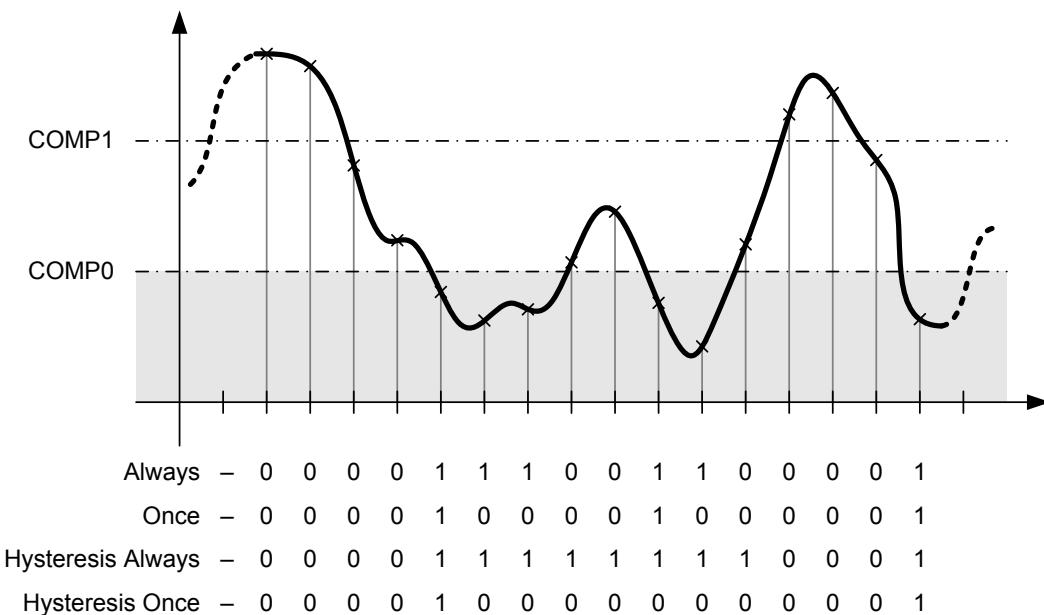
12.3.7.3 Function Ranges

The two comparison values, COMP0 and COMP1, in the **ADC Digital Comparator Range (ADCDCCMPn)** 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 **ADCDCCCTLn** 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 12-12 on page 565. 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 12-12. Low-Band Operation (CIC=0x0 and/or CTC=0x0)

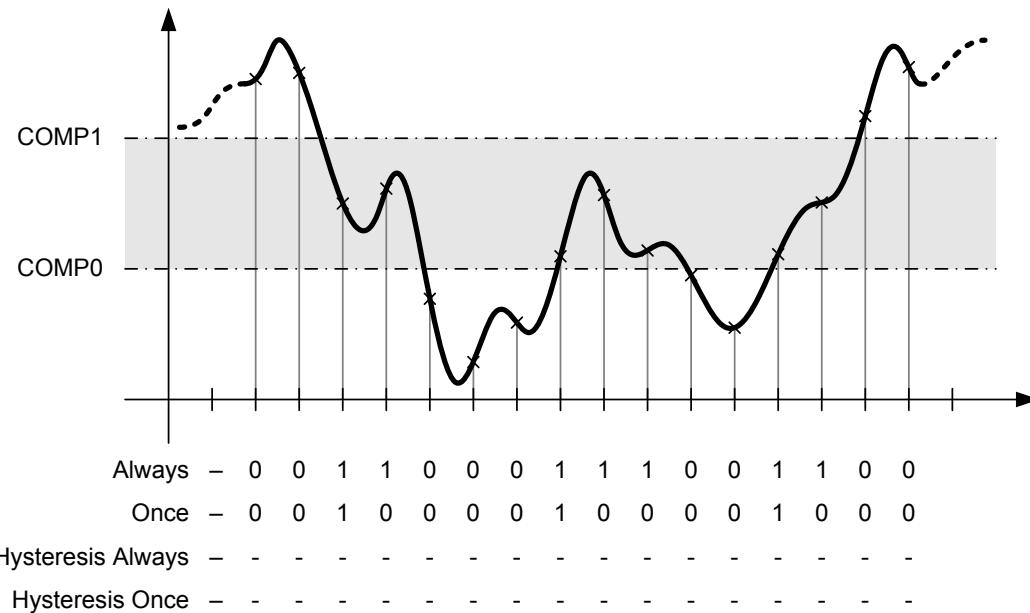


Mid-Band Operation

To operate in the mid-band region, either the CIC field or the CTC field in the **ADCDCCCTLn** register must be programmed to 0x1. This setting causes interrupts or triggers to be generated in the mid-band region according 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 12-13 on page 566. 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.

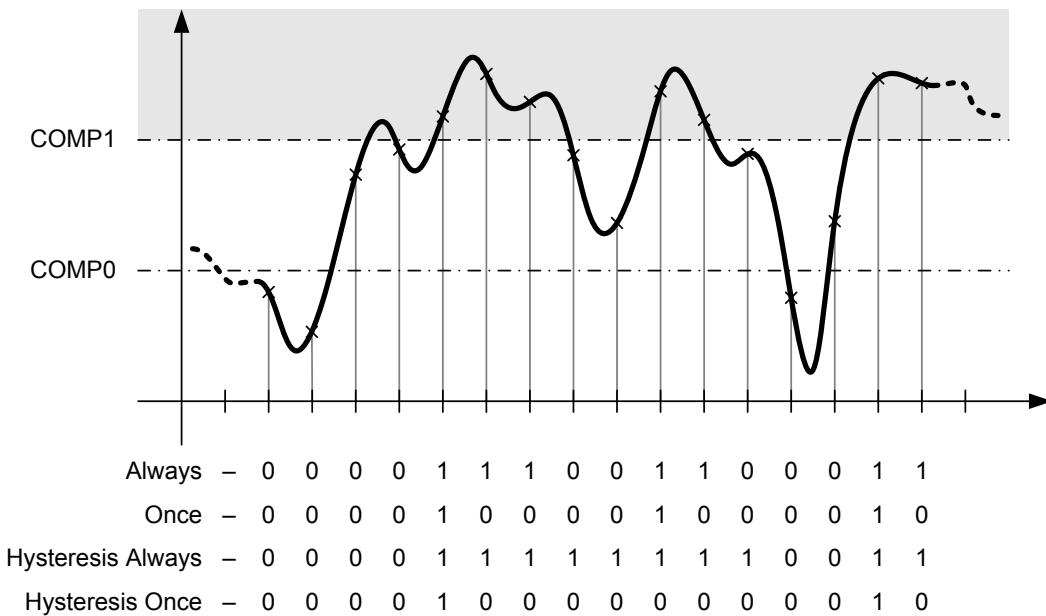
Figure 12-13. Mid-Band Operation (CIC=0x1 and/or CTC=0x1)



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 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 12-14 on page 567. 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 12-14. High-Band Operation (CIC=0x3 and/or CTC=0x3)



12.4 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 226). Using unsupported frequencies can cause faulty operation in the ADC module.

12.4.1 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 268).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register (see page 285). To find out which GPIO port to enable, refer to Table 24-5 on page 1190.
3. Set the GPIO AFSEL bits for the ADC input pins (see page 447). To determine which GPIOs to configure, see Table 24-4 on page 1182.
4. Configure the **PMCn** fields in the **GPIOPCTL** register to assign the **AINx** and **signals** to the appropriate pins (see page 465 and Table 24-5 on page 1190).
5. 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 463) in the associated GPIO block.

6. If required by the application, reconfigure the sample sequencer priorities in the **ADCSSPRI** register. The default configuration has Sample Sequencer 0 with the highest priority and Sample Sequencer 3 as the lowest priority.

12.4.2 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 **ADCSSMUXn** register.
4. For each sample in the sample sequence, configure the sample control bits in the corresponding nibble in the **ADCSSCTLn** register. When programming the last nibble, ensure that the **END** bit is set. Failure to set the **END** bit causes unpredictable behavior.
5. If interrupts are to be used, 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.

12.5 Register Map

Table 12-5 on page 568 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 268).

Table 12-5. ADC Register Map

Offset	Name	Type	Reset	Description	See page
0x000	ADCACTSS	R/W	0x0000.0000	ADC Active Sample Sequencer	571
0x004	ADCRIS	RO	0x0000.0000	ADC Raw Interrupt Status	572
0x008	ADCIM	R/W	0x0000.0000	ADC Interrupt Mask	574
0x00C	ADCISC	R/W1C	0x0000.0000	ADC Interrupt Status and Clear	576
0x010	ADCOSTAT	R/W1C	0x0000.0000	ADC Overflow Status	579
0x014	ADCEMUX	R/W	0x0000.0000	ADC Event Multiplexer Select	581

Table 12-5. ADC Register Map (*continued*)

Offset	Name	Type	Reset	Description	See page
0x018	ADCUSTAT	R/W1C	0x0000.0000	ADC Underflow Status	586
0x020	ADCSSPRI	R/W	0x0000.3210	ADC Sample Sequencer Priority	587
0x024	ADCSPC	R/W	0x0000.0000	ADC Sample Phase Control	589
0x028	ADCPSSI	R/W	-	ADC Processor Sample Sequence Initiate	590
0x030	ADCSAC	R/W	0x0000.0000	ADC Sample Averaging Control	592
0x034	ADCDCISC	R/W1C	0x0000.0000	ADC Digital Comparator Interrupt Status and Clear	593
0x038	ADCCTL	R/W	0x0000.0000	ADC Control	595
0x040	ADCSSMUX0	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 0	596
0x044	ADCSSCTL0	R/W	0x0000.0000	ADC Sample Sequence Control 0	598
0x048	ADCSSFIFO0	RO	-	ADC Sample Sequence Result FIFO 0	601
0x04C	ADCSSFSTAT0	RO	0x0000.0100	ADC Sample Sequence FIFO 0 Status	602
0x050	ADCSSOP0	R/W	0x0000.0000	ADC Sample Sequence 0 Operation	604
0x054	ADCSSDC0	R/W	0x0000.0000	ADC Sample Sequence 0 Digital Comparator Select	606
0x060	ADCSSMUX1	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 1	608
0x064	ADCSSCTL1	R/W	0x0000.0000	ADC Sample Sequence Control 1	609
0x068	ADCSSFIFO1	RO	-	ADC Sample Sequence Result FIFO 1	601
0x06C	ADCSSFSTAT1	RO	0x0000.0100	ADC Sample Sequence FIFO 1 Status	602
0x070	ADCSSOP1	R/W	0x0000.0000	ADC Sample Sequence 1 Operation	611
0x074	ADCSSDC1	R/W	0x0000.0000	ADC Sample Sequence 1 Digital Comparator Select	612
0x080	ADCSSMUX2	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 2	608
0x084	ADCSSCTL2	R/W	0x0000.0000	ADC Sample Sequence Control 2	609
0x088	ADCSSFIFO2	RO	-	ADC Sample Sequence Result FIFO 2	601
0x08C	ADCSSFSTAT2	RO	0x0000.0100	ADC Sample Sequence FIFO 2 Status	602
0x090	ADCSSOP2	R/W	0x0000.0000	ADC Sample Sequence 2 Operation	611
0x094	ADCSSDC2	R/W	0x0000.0000	ADC Sample Sequence 2 Digital Comparator Select	612
0x0A0	ADCSSMUX3	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 3	614
0x0A4	ADCSSCTL3	R/W	0x0000.0002	ADC Sample Sequence Control 3	615
0x0A8	ADCSSFIFO3	RO	-	ADC Sample Sequence Result FIFO 3	601
0x0AC	ADCSSFSTAT3	RO	0x0000.0100	ADC Sample Sequence FIFO 3 Status	602
0x0B0	ADCSSOP3	R/W	0x0000.0000	ADC Sample Sequence 3 Operation	616
0x0B4	ADCSSDC3	R/W	0x0000.0000	ADC Sample Sequence 3 Digital Comparator Select	617
0xD00	ADCDCRIC	R/W	0x0000.0000	ADC Digital Comparator Reset Initial Conditions	618

Table 12-5. ADC Register Map (*continued*)

Offset	Name	Type	Reset	Description	See page
0xE00	ADCDCCTL0	R/W	0x0000.0000	ADC Digital Comparator Control 0	623
0xE04	ADCDCCTL1	R/W	0x0000.0000	ADC Digital Comparator Control 1	623
0xE08	ADCDCCTL2	R/W	0x0000.0000	ADC Digital Comparator Control 2	623
0xE0C	ADCDCCTL3	R/W	0x0000.0000	ADC Digital Comparator Control 3	623
0xE10	ADCDCCTL4	R/W	0x0000.0000	ADC Digital Comparator Control 4	623
0xE14	ADCDCCTL5	R/W	0x0000.0000	ADC Digital Comparator Control 5	623
0xE18	ADCDCCTL6	R/W	0x0000.0000	ADC Digital Comparator Control 6	623
0xE1C	ADCDCCTL7	R/W	0x0000.0000	ADC Digital Comparator Control 7	623
0xE40	ADCDCCMP0	R/W	0x0000.0000	ADC Digital Comparator Range 0	626
0xE44	ADCDCCMP1	R/W	0x0000.0000	ADC Digital Comparator Range 1	626
0xE48	ADCDCCMP2	R/W	0x0000.0000	ADC Digital Comparator Range 2	626
0xE4C	ADCDCCMP3	R/W	0x0000.0000	ADC Digital Comparator Range 3	626
0xE50	ADCDCCMP4	R/W	0x0000.0000	ADC Digital Comparator Range 4	626
0xE54	ADCDCCMP5	R/W	0x0000.0000	ADC Digital Comparator Range 5	626
0xE58	ADCDCCMP6	R/W	0x0000.0000	ADC Digital Comparator Range 6	626
0xE5C	ADCDCCMP7	R/W	0x0000.0000	ADC Digital Comparator Range 7	626

12.6 Register Descriptions

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

Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000

This register controls the activation of the sample sequencers. Each sample sequencer can be enabled or disabled independently.

ADC Active Sample Sequencer (ADCACTSS)

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															
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	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ASEN3	R/W	0	ADC SS3 Enable
		Value	Description	
		1	Sample Sequencer 3 is enabled.	
		0	Sample Sequencer 3 is disabled.	
2	ASEN2	R/W	0	ADC SS2 Enable
		Value	Description	
		1	Sample Sequencer 2 is enabled.	
		0	Sample Sequencer 2 is disabled.	
1	ASEN1	R/W	0	ADC SS1 Enable
		Value	Description	
		1	Sample Sequencer 1 is enabled.	
		0	Sample Sequencer 1 is disabled.	
0	ASEN0	R/W	0	ADC SS0 Enable
		Value	Description	
		1	Sample Sequencer 0 is enabled.	
		0	Sample Sequencer 0 is disabled.	

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
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
		Value	Description	
		1	At least one bit in the ADCDCISC register is set, meaning that a digital comparator interrupt has occurred.	
		0	All bits in the ADCDCISC 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
		Value	Description	
		1	A sample has completed conversion and the respective ADCSSCTL3 IEn bit is set, enabling a raw interrupt.	
		0	An interrupt has not occurred.	
		This bit is cleared by writing a 1 to the IN3 bit in the ADCISC register.		
2	INR2	RO	0	SS2 Raw Interrupt Status
		Value	Description	
		1	A sample has completed conversion and the respective ADCSSCTL2 IEn bit is set, enabling a raw interrupt.	
		0	An interrupt has not occurred.	
		This bit is cleared by writing a 1 to the IN2 bit in the ADCISC register.		

Bit/Field	Name	Type	Reset	Description
1	INR1	RO	0	SS1 Raw Interrupt Status Value Description 1 A sample has completed conversion and the respective ADCSSCTL1 IEn bit is set, enabling a raw interrupt. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the IN1 bit in the ADCISC register.
0	INR0	RO	0	SS0 Raw Interrupt Status Value Description 1 A sample has completed conversion and the respective ADCSSCTL0 IEn bit is set, enabling a raw interrupt. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the IN0 bit in the ADCISC register.

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 DCONSS_n 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
	Value	Description		
	1	The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS register) is sent to the interrupt controller on the SS3 interrupt line.		
	0	The status of the digital comparators does not affect the SS3 interrupt status.		
18	DCONSS2	R/W	0	Digital Comparator Interrupt on SS2
	Value	Description		
	1	The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS register) is sent to the interrupt controller on the SS2 interrupt line.		
	0	The status of the digital comparators does not affect the SS2 interrupt status.		
17	DCONSS1	R/W	0	Digital Comparator Interrupt on SS1
	Value	Description		
	1	The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS register) is sent to the interrupt controller on the SS1 interrupt line.		
	0	The status of the digital comparators does not affect the SS1 interrupt status.		

Bit/Field	Name	Type	Reset	Description
16	DCONSS0	R/W	0	Digital Comparator Interrupt on SS0 Value Description 1 The raw interrupt signal from the digital comparators (INRDC bit in the ADCRIS register) is sent to the interrupt controller on the SS0 interrupt line. 0 The status of the digital comparators does not affect the SS0 interrupt status.
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	MASK3	R/W	0	SS3 Interrupt Mask Value Description 1 The raw interrupt signal from Sample Sequencer 3 (ADCRIS register INR3 bit) is sent to the interrupt controller. 0 The status of Sample Sequencer 3 does not affect the SS3 interrupt status.
2	MASK2	R/W	0	SS2 Interrupt Mask Value Description 1 The raw interrupt signal from Sample Sequencer 2 (ADCRIS register INR2 bit) is sent to the interrupt controller. 0 The status of Sample Sequencer 2 does not affect the SS2 interrupt status.
1	MASK1	R/W	0	SS1 Interrupt Mask Value Description 1 The raw interrupt signal from Sample Sequencer 1 (ADCRIS register INR1 bit) is sent to the interrupt controller. 0 The status of Sample Sequencer 1 does not affect the SS1 interrupt status.
0	MASK0	R/W	0	SS0 Interrupt Mask Value Description 1 The raw interrupt signal from Sample Sequencer 0 (ADCRIS register INR0 bit) is sent to the interrupt controller. 0 The status of Sample Sequencer 0 does not affect the SS0 interrupt status.

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 INRn bits are still cleared via the **ADCISC** register, even if the INn 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
	Value	Description		
	1	Both the INRDC bit in the ADCRIS register and the DCONSS3 bit in the ADCIM register are set, providing a level-base interrupt to the interrupt controller.		
	0	No interrupt has occurred or the interrupt is masked.		
	This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the ADCRIS register.			
18	DCINSS2	RO	0	Digital Comparator Interrupt Status on SS2
	Value	Description		
	1	Both the INRDC bit in the ADCRIS register and the DCONSS2 bit in the ADCIM register are set, providing a level-base interrupt to the interrupt controller.		
	0	No interrupt has occurred or the interrupt is masked.		
	This bit is cleared by writing a 1 to it. Clearing this bit also clears the INRDC bit in the ADCRIS register.			

Bit/Field	Name	Type	Reset	Description
17	DCINSS1	RO	0	Digital Comparator Interrupt Status on SS1 Value Description 1 Both the <code>INRDC</code> bit in the ADCRIS register and the <code>DCONSS1</code> bit in the ADCIM register are set, providing a level-base interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1 to it. Clearing this bit also clears the <code>INRDC</code> bit in the ADCRIS register.
16	DCINSS0	RO	0	Digital Comparator Interrupt Status on SS0 Value Description 1 Both the <code>INRDC</code> bit in the ADCRIS register and the <code>DCONSS0</code> bit in the ADCIM register are set, providing a level-base interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1 to it. Clearing this bit also clears the <code>INRDC</code> bit in the ADCRIS register.
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	IN3	R/W1C	0	SS3 Interrupt Status and Clear Value Description 1 Both the <code>INR3</code> bit in the ADCRIS register and the <code>MASK3</code> bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the <code>INR3</code> bit in the ADCRIS register.
2	IN2	R/W1C	0	SS2 Interrupt Status and Clear Value Description 1 Both the <code>INR2</code> bit in the ADCRIS register and the <code>MASK2</code> bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the <code>INR2</code> bit in the ADCRIS register.

Bit/Field	Name	Type	Reset	Description
1	IN1	R/W1C	0	SS1 Interrupt Status and Clear Value Description 1 Both the INR1 bit in the ADCRIS register and the MASK1 bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the INR1 bit in the ADCRIS register.
0	IN0	R/W1C	0	SS0 Interrupt Status and Clear Value Description 1 Both the INR0 bit in the ADCRIS register and the MASK0 bit in the ADCIM register are set, providing a level-based interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the INR0 bit in the ADCRIS register.

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															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OV3	R/W1C	0	SS3 FIFO Overflow
	Value	Description		
	1	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.		
	0	The FIFO has not overflowed.		
	This bit is cleared by writing a 1.			
2	OV2	R/W1C	0	SS2 FIFO Overflow
	Value	Description		
	1	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.		
	0	The FIFO has not overflowed.		
	This bit is cleared by writing a 1.			
1	OV1	R/W1C	0	SS1 FIFO Overflow
	Value	Description		
	1	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.		
	0	The FIFO has not overflowed.		
	This bit is cleared by writing a 1.			

Bit/Field	Name	Type	Reset	Description
0	OV0	R/W1C	0	SS0 FIFO Overflow
Value Description				
		1		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.
		0		The FIFO has not overflowed.
This bit is cleared by writing a 1.				

Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014

The **ADCEMUX** selects the event (trigger) that initiates sampling for each sample sequencer. Each sample sequencer can be configured with a unique trigger source.

ADC Event Multiplexer Select (ADCEMUX)

ADC0 base: 0x4003.8000

ADC1 base: 0x4003.9000

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
	EM3				EM2				EM1				EM0			
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.

Bit/Field	Name	Type	Reset	Description												
15:12	EM3	R/W	0x0	<p>SS3 Trigger Select</p> <p>This field selects the trigger source for Sample Sequencer 3.</p> <p>The valid configurations for this field are:</p> <table> <thead> <tr> <th>Value</th><th>Event</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>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>reserved</td></tr> <tr> <td>0x4</td><td>External (GPIO PB4)</td></tr> </tbody> </table> <p>Note: PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.</p>	Value	Event	0x0	Processor (default)	0x1	Analog Comparator 0	0x2	Analog Comparator 1	0x3	reserved	0x4	External (GPIO PB4)
Value	Event															
0x0	Processor (default)															
0x1	Analog Comparator 0															
0x2	Analog Comparator 1															
0x3	reserved															
0x4	External (GPIO PB4)															
		0x5		Timer												
				In addition, the trigger must be enabled with the TnOTE bit in the GPTMCTL register (see page 500).												
		0x6		PWM0												
				The PWM module 0 trigger can be configured with the PWM0 Interrupt and Trigger Enable (PWM0INTEN) register, see page 1105.												
		0x7		PWM1												
				The PWM module 1 trigger can be configured with the PWM1INTEN register, see page 1105.												
		0x8		PWM2												
				The PWM module 2 trigger can be configured with the PWM2INTEN register, see page 1105.												
		0x9		PWM3												
				The PWM module 3 trigger can be configured with the PWM3INTEN register, see page 1105.												
		0xA-0xE		reserved												
		0xF		Always (continuously sample)												

Bit/Field	Name	Type	Reset	Description
11:8	EM2	R/W	0x0	SS2 Trigger Select This field selects the trigger source for Sample Sequencer 2. The valid configurations for this field are:
			Value	Event
			0x0	Processor (default)
			0x1	Analog Comparator 0
			0x2	Analog Comparator 1
			0x3	reserved
			0x4	External (GPIO PB4) Note: 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 500).
			0x6	PWM0 The PWM module 0 trigger can be configured with the PWM0 Interrupt and Trigger Enable (PWM0INTEN) register, see page 1105.
			0x7	PWM1 The PWM module 1 trigger can be configured with the PWM1INTEN register, see page 1105.
			0x8	PWM2 The PWM module 2 trigger can be configured with the PWM2INTEN register, see page 1105.
			0x9	PWM3 The PWM module 3 trigger can be configured with the PWM3INTEN register, see page 1105.
			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> <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>reserved</td></tr> <tr> <td>0x4</td><td>External (GPIO PB4)</td></tr> </tbody> </table> <p>Note: PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.</p>	Value	Event	0x0	Processor (default)	0x1	Analog Comparator 0	0x2	Analog Comparator 1	0x3	reserved	0x4	External (GPIO PB4)
Value	Event															
0x0	Processor (default)															
0x1	Analog Comparator 0															
0x2	Analog Comparator 1															
0x3	reserved															
0x4	External (GPIO PB4)															
			0x5	Timer												
				In addition, the trigger must be enabled with the TnOTE bit in the GPTMCTL register (see page 500).												
			0x6	PWM0												
				The PWM module 0 trigger can be configured with the PWM0 Interrupt and Trigger Enable (PWM0INTEN) register, see page 1105.												
			0x7	PWM1												
				The PWM module 1 trigger can be configured with the PWM1INTEN register, see page 1105.												
			0x8	PWM2												
				The PWM module 2 trigger can be configured with the PWM2INTEN register, see page 1105.												
			0x9	PWM3												
				The PWM module 3 trigger can be configured with the PWM3INTEN register, see page 1105.												
			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> <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>reserved</td></tr> <tr> <td>0x4</td><td>External (GPIO PB4)</td></tr> </tbody> </table> <p>Note: PB4 can be used to trigger the ADC. However, the PB4/AIN10 pin cannot be used as both a GPIO and an analog input.</p>	Value	Event	0x0	Processor (default)	0x1	Analog Comparator 0	0x2	Analog Comparator 1	0x3	reserved	0x4	External (GPIO PB4)
Value	Event															
0x0	Processor (default)															
0x1	Analog Comparator 0															
0x2	Analog Comparator 1															
0x3	reserved															
0x4	External (GPIO PB4)															
			0x5	Timer												
				In addition, the trigger must be enabled with the TnOTE bit in the GPTMCTL register (see page 500).												
			0x6	PWM0												
				The PWM module 0 trigger can be configured with the PWM0 Interrupt and Trigger Enable (PWM0INTEN) register, see page 1105.												
			0x7	PWM1												
				The PWM module 1 trigger can be configured with the PWM1INTEN register, see page 1105.												
			0x8	PWM2												
				The PWM module 2 trigger can be configured with the PWM2INTEN register, see page 1105.												
			0x9	PWM3												
				The PWM module 3 trigger can be configured with the PWM3INTEN register, see page 1105.												
			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

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	UV3	R/W1C	0	SS3 FIFO Underflow The valid configurations for this field are shown below. This bit is cleared by writing a 1.
	Value	Description		
	1	The FIFO for the Sample Sequencer 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.		
	0	The FIFO has not underflowed.		
2	UV2	R/W1C	0	SS2 FIFO Underflow The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.
1	UV1	R/W1C	0	SS1 FIFO Underflow The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.
0	UV0	R/W1C	0	SS0 FIFO Underflow The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.

Register 8: ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020

This register sets the priority for each of the sample sequencers. Out of reset, Sequencer 0 has the highest priority, and 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												
Reset	0	0	0	0	0	0	0	0	0	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												
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	<p>SS0 Priority</p> <p>This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 0. A priority encoding of 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.</p>

Register 9: ADC Sample Phase Control (ADCSPC), offset 0x024

This register allows the ADC module to sample at one of 16 different discrete phases from 0.0° through 337.5°. For example, the sample rate could be effectively doubled by sampling a signal using one ADC module configured with the standard sample time and the second ADC module configured with a 180.0° phase lag.

Note: Care should be taken when the PHASE field is non-zero, as the resulting delay in sampling the AIN_x input may result in undesirable system consequences. Designers should carefully consider the impact of this delay.

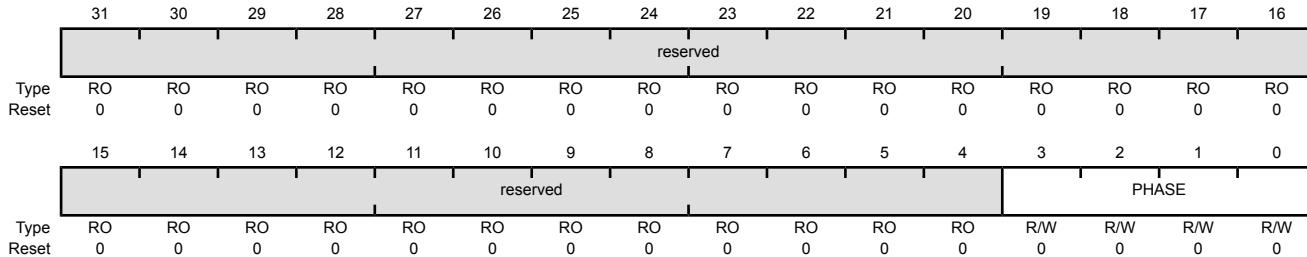
ADC Sample Phase Control (ADCSPC)

ADC0 base: 0x4003.8000

ADC1 base: 0x4003.9000

Offset 0x024

Type R/W, 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.																																		
3:0	PHASE	R/W	0x0	<p>Phase Difference This field selects the sample phase difference from the standard sample time.</p> <table border="0"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>ADC sample lags by 0.0°</td> </tr> <tr> <td>0x1</td> <td>ADC sample lags by 22.5°</td> </tr> <tr> <td>0x2</td> <td>ADC sample lags by 45.0°</td> </tr> <tr> <td>0x3</td> <td>ADC sample lags by 67.5°</td> </tr> <tr> <td>0x4</td> <td>ADC sample lags by 90.0°</td> </tr> <tr> <td>0x5</td> <td>ADC sample lags by 112.5°</td> </tr> <tr> <td>0x6</td> <td>ADC sample lags by 135.0°</td> </tr> <tr> <td>0x7</td> <td>ADC sample lags by 157.5°</td> </tr> <tr> <td>0x8</td> <td>ADC sample lags by 180.0°</td> </tr> <tr> <td>0x9</td> <td>ADC sample lags by 202.5°</td> </tr> <tr> <td>0xA</td> <td>ADC sample lags by 225.0°</td> </tr> <tr> <td>0xB</td> <td>ADC sample lags by 247.5°</td> </tr> <tr> <td>0xC</td> <td>ADC sample lags by 270.0°</td> </tr> <tr> <td>0xD</td> <td>ADC sample lags by 292.5°</td> </tr> <tr> <td>0xE</td> <td>ADC sample lags by 315.0°</td> </tr> <tr> <td>0xF</td> <td>ADC sample lags by 337.5°</td> </tr> </tbody> </table>	Value	Description	0x0	ADC sample lags by 0.0°	0x1	ADC sample lags by 22.5°	0x2	ADC sample lags by 45.0°	0x3	ADC sample lags by 67.5°	0x4	ADC sample lags by 90.0°	0x5	ADC sample lags by 112.5°	0x6	ADC sample lags by 135.0°	0x7	ADC sample lags by 157.5°	0x8	ADC sample lags by 180.0°	0x9	ADC sample lags by 202.5°	0xA	ADC sample lags by 225.0°	0xB	ADC sample lags by 247.5°	0xC	ADC sample lags by 270.0°	0xD	ADC sample lags by 292.5°	0xE	ADC sample lags by 315.0°	0xF	ADC sample lags by 337.5°
Value	Description																																					
0x0	ADC sample lags by 0.0°																																					
0x1	ADC sample lags by 22.5°																																					
0x2	ADC sample lags by 45.0°																																					
0x3	ADC sample lags by 67.5°																																					
0x4	ADC sample lags by 90.0°																																					
0x5	ADC sample lags by 112.5°																																					
0x6	ADC sample lags by 135.0°																																					
0x7	ADC sample lags by 157.5°																																					
0x8	ADC sample lags by 180.0°																																					
0x9	ADC sample lags by 202.5°																																					
0xA	ADC sample lags by 225.0°																																					
0xB	ADC sample lags by 247.5°																																					
0xC	ADC sample lags by 270.0°																																					
0xD	ADC sample lags by 292.5°																																					
0xE	ADC sample lags by 315.0°																																					
0xF	ADC sample lags by 337.5°																																					

Register 10: 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 R/W, reset -

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

Bit/Field	Name	Type	Reset	Description
31	GSYNC	R/W	0	Global Synchronize
				Value Description
			1	This bit initiates sampling in multiple ADC modules at the same time. Any ADC module that has been initialized by setting an SS _n bit and the SYNCWAIT bit starts sampling once this bit is written.
			0	This bit is cleared once sampling has been initiated.
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	SYNCWAIT	R/W	0	Synchronize Wait
				Value Description
			1	This bit allows the sample sequences to be initiated, but delays sampling until the GSYNC bit is set.
			0	Sampling begins when a sample sequence has been initiated.
26:4	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.

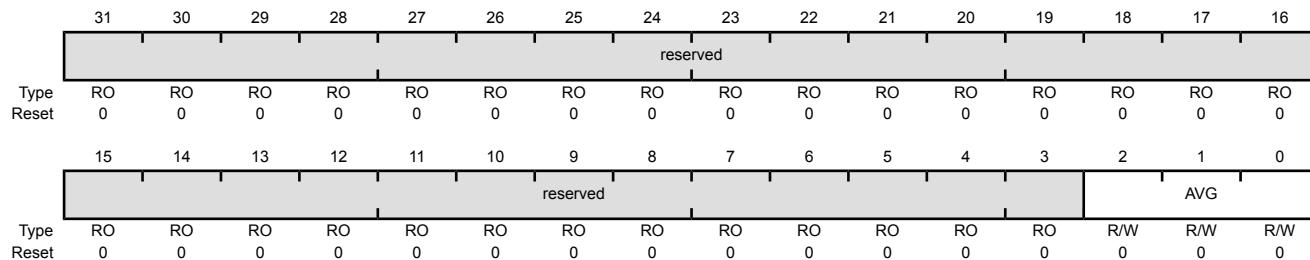
Bit/Field	Name	Type	Reset	Description						
3	SS3	WO	-	<p>SS3 Initiate</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Begin sampling on Sample Sequencer 3, if the sequencer is enabled in the ADCACTSS register.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table> <p>Only a write by software is valid; a read of this register returns no meaningful data.</p>	Value	Description	1	Begin sampling on Sample Sequencer 3, if the sequencer is enabled in the ADCACTSS register.	0	No effect.
Value	Description									
1	Begin sampling on Sample Sequencer 3, if the sequencer is enabled in the ADCACTSS register.									
0	No effect.									
2	SS2	WO	-	<p>SS2 Initiate</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Begin sampling on Sample Sequencer 2, if the sequencer is enabled in the ADCACTSS register.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table> <p>Only a write by software is valid; a read of this register returns no meaningful data.</p>	Value	Description	1	Begin sampling on Sample Sequencer 2, if the sequencer is enabled in the ADCACTSS register.	0	No effect.
Value	Description									
1	Begin sampling on Sample Sequencer 2, if the sequencer is enabled in the ADCACTSS register.									
0	No effect.									
1	SS1	WO	-	<p>SS1 Initiate</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Begin sampling on Sample Sequencer 1, if the sequencer is enabled in the ADCACTSS register.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table> <p>Only a write by software is valid; a read of this register returns no meaningful data.</p>	Value	Description	1	Begin sampling on Sample Sequencer 1, if the sequencer is enabled in the ADCACTSS register.	0	No effect.
Value	Description									
1	Begin sampling on Sample Sequencer 1, if the sequencer is enabled in the ADCACTSS register.									
0	No effect.									
0	SS0	WO	-	<p>SS0 Initiate</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Begin sampling on Sample Sequencer 0, if the sequencer is enabled in the ADCACTSS register.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table> <p>Only a write by software is valid; a read of this register returns no meaningful data.</p>	Value	Description	1	Begin sampling on Sample Sequencer 0, if the sequencer is enabled in the ADCACTSS register.	0	No effect.
Value	Description									
1	Begin sampling on Sample Sequencer 0, if the sequencer is enabled in the ADCACTSS register.									
0	No effect.									

Register 11: ADC Sample Averaging Control (ADCSAC), offset 0x030

This register controls the amount of hardware averaging applied to conversion results. The final conversion result stored in the FIFO is averaged from 2^{AVG} consecutive ADC samples at the specified ADC speed. If AVG is 0, the sample is passed directly through without any averaging. If AVG=6, then 64 consecutive ADC samples are averaged to generate one result in the sequencer FIFO. An AVG=7 provides unpredictable results.

ADC Sample Averaging Control (ADCSAC)

ADC0 base: 0x4003.8000
 ADC1 base: 0x4003.9000
 Offset 0x030
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description																		
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																		
2:0	AVG	R/W	0x0	<p>Hardware Averaging Control</p> <p>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.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>No hardware oversampling</td> </tr> <tr> <td>0x1</td> <td>2x hardware oversampling</td> </tr> <tr> <td>0x2</td> <td>4x hardware oversampling</td> </tr> <tr> <td>0x3</td> <td>8x hardware oversampling</td> </tr> <tr> <td>0x4</td> <td>16x hardware oversampling</td> </tr> <tr> <td>0x5</td> <td>32x hardware oversampling</td> </tr> <tr> <td>0x6</td> <td>64x hardware oversampling</td> </tr> <tr> <td>0x7</td> <td>reserved</td> </tr> </tbody> </table>	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
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 12: 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.

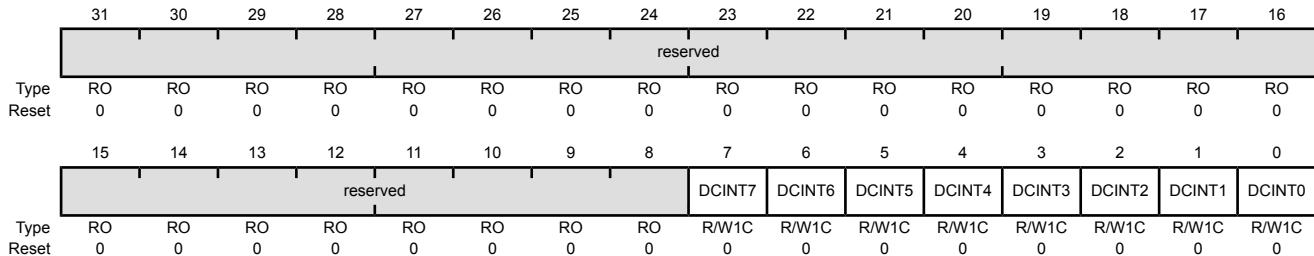
ADC Digital Comparator Interrupt Status and Clear (ADCDCISC)

ADC0 base: 0x4003.8000

ADC1 base: 0x4003.9000

Offset 0x034

Type R/W1C, reset 0x0000.0000



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 Value Description 1 Digital Comparator 7 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
6	DCINT6	R/W1C	0	Digital Comparator 6 Interrupt Status and Clear Value Description 1 Digital Comparator 6 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
5	DCINT5	R/W1C	0	Digital Comparator 5 Interrupt Status and Clear Value Description 1 Digital Comparator 5 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.

Bit/Field	Name	Type	Reset	Description
4	DCINT4	R/W1C	0	Digital Comparator 4 Interrupt Status and Clear Value Description 1 Digital Comparator 4 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
3	DCINT3	R/W1C	0	Digital Comparator 3 Interrupt Status and Clear Value Description 1 Digital Comparator 3 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
2	DCINT2	R/W1C	0	Digital Comparator 2 Interrupt Status and Clear Value Description 1 Digital Comparator 2 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
1	DCINT1	R/W1C	0	Digital Comparator 1 Interrupt Status and Clear Value Description 1 Digital Comparator 1 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
0	DCINT0	R/W1C	0	Digital Comparator 0 Interrupt Status and Clear Value Description 1 Digital Comparator 0 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.

Register 13: ADC Control (ADCCTL), offset 0x038

This register configures the voltage reference. The voltage reference for the conversion can be the internal 3.0-V reference or an external voltage reference in the range of 2.4 V to 3.06 V.

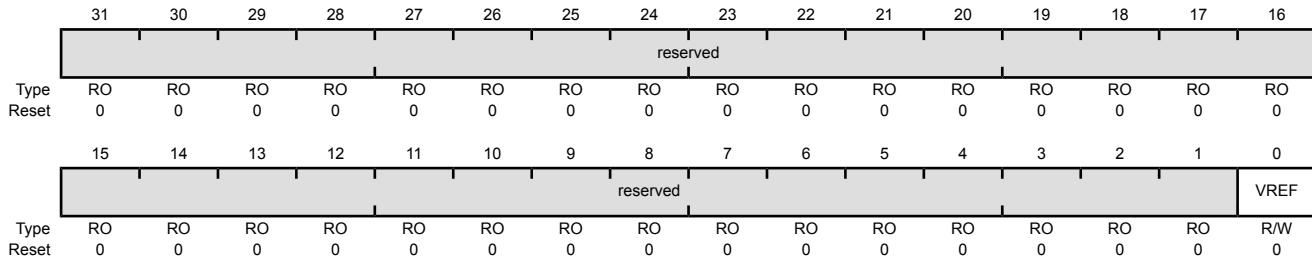
ADC Control (ADCCTL)

ADC0 base: 0x4003.8000

ADC1 base: 0x4003.9000

Offset 0x038

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	VREF	R/W	0	Voltage Reference Select Value Description 1 The external VREFA input is the voltage reference. 0 The internal reference as the voltage reference.

Register 14: 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												
Reset	0	0	0	0	0	0	0	0	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												
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 15: 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
Type	TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4
Reset	R/W 0															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Reset	R/W 0															

Bit/Field	Name	Type	Reset	Description						
31	TS7	R/W	0	8th Sample Temp Sensor Select						
				<table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>The temperature sensor is read during the eighth sample of the sample sequence.</td> </tr> <tr> <td>0</td> <td>The input pin specified by the ADCSSMUXn register is read during the eighth sample of the sample sequence.</td> </tr> </tbody> </table>	Value	Description	1	The temperature sensor is read during the eighth sample of the sample sequence.	0	The input pin specified by the ADCSSMUXn register is read during the eighth sample of the sample sequence.
Value	Description									
1	The temperature sensor is read during the eighth sample of the sample sequence.									
0	The input pin specified by the ADCSSMUXn register is read during the eighth sample of the sample sequence.									
30	IE7	R/W	0	8th Sample Interrupt Enable						
				<table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>The raw interrupt signal (INR0 bit) is asserted at the end of the eighth sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller.</td> </tr> <tr> <td>0</td> <td>The raw interrupt is not asserted to the interrupt controller.</td> </tr> </tbody> </table> <p>It is legal to have multiple samples within a sequence generate interrupts.</p>	Value	Description	1	The raw interrupt signal (INR0 bit) is asserted at the end of the eighth sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller.	0	The raw interrupt is not asserted to the interrupt controller.
Value	Description									
1	The raw interrupt signal (INR0 bit) is asserted at the end of the eighth sample's conversion. If the MASK0 bit in the ADCIM register is set, the interrupt is promoted to the interrupt controller.									
0	The raw interrupt is not asserted to the interrupt controller.									
29	END7	R/W	0	8th Sample is End of Sequence						
				<table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>The eighth sample is the last sample of the sequence.</td> </tr> <tr> <td>0</td> <td>Another sample in the sequence is the final sample.</td> </tr> </tbody> </table> <p>It is possible to end the sequence on any sample position. Software must set an ENDn bit somewhere within the sequence. Samples defined after the sample containing a set ENDn bit are not requested for conversion even though the fields may be non-zero.</p>	Value	Description	1	The eighth sample is the last sample of the sequence.	0	Another sample in the sequence is the final sample.
Value	Description									
1	The eighth sample is the last sample of the sequence.									
0	Another sample in the sequence is the final sample.									

Bit/Field	Name	Type	Reset	Description
28	D7	R/W	0	8th Sample Diff Input Select Value Description 1 The analog input is differentially sampled. The corresponding ADCSSMUXn nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". 0 The analog inputs are not differentially sampled. Because the temperature sensor does not have a differential option, this bit must not be set when the TS7 bit is set.
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.

Bit/Field	Name	Type	Reset	Description
12	D3	R/W	0	4th Sample Diff Input Select Same definition as D7 but used during the fourth sample.
11	TS2	R/W	0	3rd Sample Temp Sensor Select Same definition as TS7 but used during the third sample.
10	IE2	R/W	0	3rd Sample Interrupt Enable Same definition as IE7 but used during the third sample.
9	END2	R/W	0	3rd Sample is End of Sequence Same definition as END7 but used during the third sample.
8	D2	R/W	0	3rd Sample Diff Input Select Same definition as D7 but used during the third sample.
7	TS1	R/W	0	2nd Sample Temp Sensor Select Same definition as TS7 but used during the second sample.
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 16: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048**Register 17: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068****Register 18: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088****Register 19: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8**

Important: Use caution when reading this register. Performing a read may change bit status.

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.

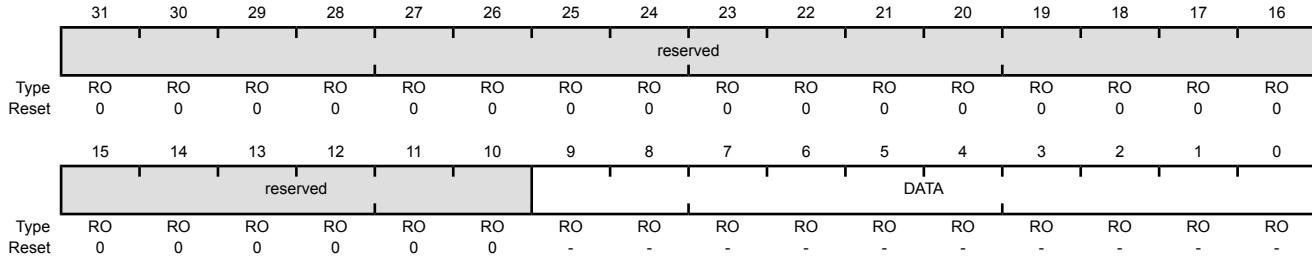
ADC Sample Sequence Result FIFO n (ADCSSFIFO n)

ADC0 base: 0x4003.8000

ADC1 base: 0x4003.9000

Offset 0x048

Type RO, reset -



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.
9:0	DATA	RO	-	Conversion Result Data

Register 20: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C**Register 21: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C****Register 22: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C****Register 23: 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
				Value Description
				1 The FIFO is currently full.
				0 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
				Value Description
				1 The FIFO is currently empty.
				0 The FIFO is not currently empty.

Bit/Field	Name	Type	Reset	Description
7:4	HPTR	RO	0x0	FIFO Head Pointer This field contains the current "head" pointer index for the FIFO, that is, the next entry to be written.
3:0	TPTR	RO	0x0	FIFO Tail Pointer This field contains the current "tail" pointer index for the FIFO, that is, the next entry to be read.

Register 24: 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												
Reset	0	0	0	0	0	0	0	0	0	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												
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
	Value	Description		
	1	The eighth sample is sent to the digital comparator unit specified by the S7DCSEL bit in the ADCSSDC0 register, and the value is not written to the FIFO.		
	0	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 S7DCOP 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 S7DCOP 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 S7DCOP 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 S7DCOP 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 S7DCOP 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 S7DCOP 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 S7DCOP but used during the first sample.

Register 25: 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												
Reset	0	0	0	0	0	0	0	0	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												
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	Sample 7 Digital Comparator Select When the S ₇ DCOP bit in the ADCSSOP0 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. Note: Values not listed are reserved.
				Value Description 0x0 Digital Comparator Unit 0 (ADCDCCMP0 and ADCCCTL0) 0x1 Digital Comparator Unit 1 (ADCDCCMP1 and ADCCCTL1) 0x2 Digital Comparator Unit 2 (ADCDCCMP2 and ADCCCTL2) 0x3 Digital Comparator Unit 3 (ADCDCCMP3 and ADCCCTL3) 0x4 Digital Comparator Unit 4 (ADCDCCMP4 and ADCCCTL4) 0x5 Digital Comparator Unit 5 (ADCDCCMP5 and ADCCCTL5) 0x6 Digital Comparator Unit 6 (ADCDCCMP6 and ADCCCTL6) 0x7 Digital Comparator Unit 7 (ADCDCCMP7 and ADCCCTL7)
27:24	S6DCSEL	R/W	0x0	Sample 6 Digital Comparator Select This field has the same encodings as S ₇ DCSEL but is used during the seventh sample.
23:20	S5DCSEL	R/W	0x0	Sample 5 Digital Comparator Select This field has the same encodings as S ₇ DCSEL but is used during the sixth sample.
19:16	S4DCSEL	R/W	0x0	Sample 4 Digital Comparator Select This field has the same encodings as S ₇ DCSEL but is used during the fifth sample.
15:12	S3DCSEL	R/W	0x0	Sample 3 Digital Comparator Select This field has the same encodings as S ₇ DCSEL but is used during the fourth sample.

Bit/Field	Name	Type	Reset	Description
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 26: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060

Register 27: 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 596 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 28: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064**Register 29: 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 598 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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Reset	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 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	TS3	R/W	0	4th Sample Temp Sensor Select Same definition as TS7 but used during the fourth sample.
14	IE3	R/W	0	4th Sample Interrupt Enable Same definition as IE7 but used during the fourth sample.
13	END3	R/W	0	4th Sample is End of Sequence Same definition as END7 but used during the fourth sample.
12	D3	R/W	0	4th Sample Diff Input Select Same definition as D7 but used during the fourth sample.
11	TS2	R/W	0	3rd Sample Temp Sensor Select Same definition as TS7 but used during the third sample.
10	IE2	R/W	0	3rd Sample Interrupt Enable Same definition as IE7 but used during the third sample.
9	END2	R/W	0	3rd Sample is End of Sequence Same definition as END7 but used during the third sample.
8	D2	R/W	0	3rd Sample Diff Input Select Same definition as D7 but used during the third sample.
7	TS1	R/W	0	2nd Sample Temp Sensor Select Same definition as TS7 but used during the second sample.

Bit/Field	Name	Type	Reset	Description
6	IE1	R/W	0	2nd Sample Interrupt Enable Same definition as IE7 but used during the second sample.
5	END1	R/W	0	2nd Sample is End of Sequence Same definition as END7 but used during the second sample.
4	D1	R/W	0	2nd Sample Diff Input Select Same definition as D7 but used during the second sample.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as TS7 but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as IE7 but used during the first sample.
1	END0	R/W	0	1st Sample is End of Sequence Same definition as END7 but used during the first sample.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as D7 but used during the first sample.

Register 30: ADC Sample Sequence 1 Operation (ADCSSOP1), offset 0x070**Register 31: 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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Type	reserved			S3DCOP	reserved			S2DCOP	reserved			S1DCOP	reserved		S0DCOP
Reset	RO	RO	RO	R/W	RO	RO	RO	R/W	RO	RO	RO	R/W	RO	RO	R/W

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	S3DCOP	R/W	0	Sample 3 Digital Comparator Operation
	Value	Description		
	1	The fourth sample is sent to the digital comparator unit specified by the S3DCSEL bit in the ADCSSDCOn register, and the value is not written to the FIFO.		
	0	The fourth sample is saved in Sample Sequence FIFOOn.		
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 32: ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1), offset 0x074

Register 33: 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 $S_{n}DCOP$ 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

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

31:16 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 S3DCSEL R/W 0x0 Sample 3 Digital Comparator Select
When the $S_{n}DCOP$ bit in the **ADCSSOPn** 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.

Note: Values not listed are reserved.

Value	Description
0x0	Digital Comparator Unit 0 (ADCDCCMP0 and ADCCCTL0)
0x1	Digital Comparator Unit 1 (ADCDCCMP1 and ADCCCTL1)
0x2	Digital Comparator Unit 2 (ADCDCCMP2 and ADCCCTL2)
0x3	Digital Comparator Unit 3 (ADCDCCMP3 and ADCCCTL3)
0x4	Digital Comparator Unit 4 (ADCDCCMP4 and ADCCCTL4)
0x5	Digital Comparator Unit 5 (ADCDCCMP5 and ADCCCTL5)
0x6	Digital Comparator Unit 6 (ADCDCCMP6 and ADCCCTL6)
0x7	Digital Comparator Unit 7 (ADCDCCMP7 and ADCCCTL7)

11:8 S2DCSEL R/W 0x0 Sample 2 Digital Comparator Select
This field has the same encodings as **S3DCSEL** but is used during the third sample.

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 34: 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 596 for detailed bit descriptions.

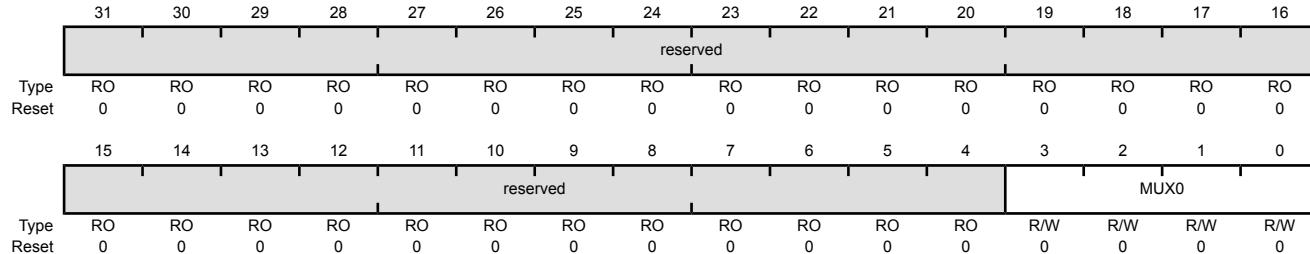
ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3)

ADC0 base: 0x4003.8000

ADC1 base: 0x4003.9000

Offset 0x0A0

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	MUX0	R/W	0	1st Sample Input Select

Register 35: 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 598 for detailed bit descriptions.

ADC Sample Sequence Control 3 (ADCSSCTL3)

ADC0 base: 0x4003.8000

ADC1 base: 0x4003.9000

Offset 0xA4

Type R/W, reset 0x0000.0002

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as 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	1	1st Sample is End of Sequence Same definition as END7 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 D7 but used during the first sample.

Register 36: 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

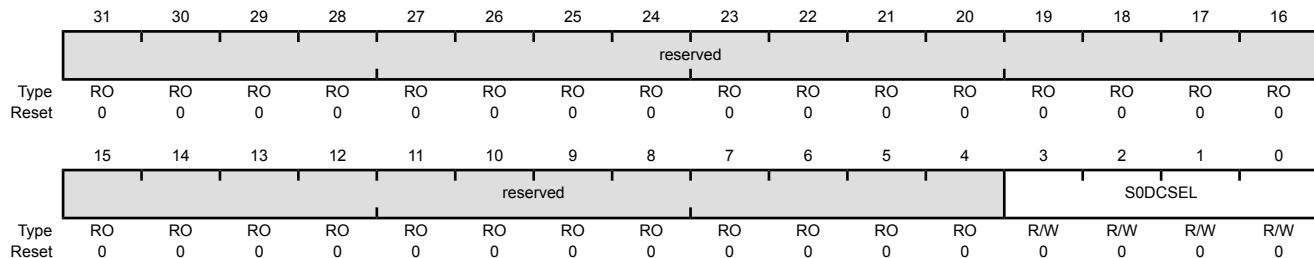
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	S0DCOP	R/W	0	Sample 0 Digital Comparator Operation
		Value	Description	
		1	The sample is sent to the digital comparator unit specified by the S0DCSEL bit in the ADCSSDC03 register, and the value is not written to the FIFO.	
		0	The sample is saved in Sample Sequence FIFO3.	

Register 37: 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	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	S0DCSEL	R/W	0x0	Sample 0 Digital Comparator Select When the S ₀ DCOP 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 (ADCDCCMP0 and ADCCCTL0)
0x1	Digital Comparator Unit 1 (ADCDCCMP1 and ADCCCTL1)
0x2	Digital Comparator Unit 2 (ADCDCCMP2 and ADCCCTL2)
0x3	Digital Comparator Unit 3 (ADCDCCMP3 and ADCCCTL3)
0x4	Digital Comparator Unit 4 (ADCDCCMP4 and ADCCCTL4)
0x5	Digital Comparator Unit 5 (ADCDCCMP5 and ADCCCTL5)
0x6	Digital Comparator Unit 6 (ADCDCCMP6 and ADCCCTL6)
0x7	Digital Comparator Unit 7 (ADCDCCMP7 and ADCCCTL7)

Register 38: 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	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
	reserved								DCINT7	DCINT6	DCINT5	DCINT4	DCINT3	DCINT2	DCINT1	DCINT0
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						
31:24	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	<p>Digital Comparator Trigger 7</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Resets the Digital Comparator 7 trigger unit to its initial conditions.</td> </tr> <tr> <td>0</td> <td>No effect.</td> </tr> </tbody> </table> <p>When the trigger has been cleared, this bit is automatically cleared. 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.</p>	Value	Description	1	Resets the Digital Comparator 7 trigger unit to its initial conditions.	0	No effect.
Value	Description									
1	Resets the Digital Comparator 7 trigger unit to its initial conditions.									
0	No effect.									
22	DCTRIG6	R/W	0	<p>Digital Comparator Trigger 6</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Resets the Digital Comparator 6 trigger unit to its initial conditions.</td> </tr> <tr> <td>0</td> <td>No effect.</td> </tr> </tbody> </table> <p>When the trigger has been cleared, this bit is automatically cleared. 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.</p>	Value	Description	1	Resets the Digital Comparator 6 trigger unit to its initial conditions.	0	No effect.
Value	Description									
1	Resets the Digital Comparator 6 trigger unit to its initial conditions.									
0	No effect.									

Bit/Field	Name	Type	Reset	Description						
21	DCTRIG5	R/W	0	<p>Digital Comparator Trigger 5</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Resets the Digital Comparator 5 trigger unit to its initial conditions.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table> <p>When the trigger has been cleared, this bit is automatically cleared. 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.</p>	Value	Description	1	Resets the Digital Comparator 5 trigger unit to its initial conditions.	0	No effect.
Value	Description									
1	Resets the Digital Comparator 5 trigger unit to its initial conditions.									
0	No effect.									
20	DCTRIG4	R/W	0	<p>Digital Comparator Trigger 4</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Resets the Digital Comparator 4 trigger unit to its initial conditions.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table> <p>When the trigger has been cleared, this bit is automatically cleared. 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.</p>	Value	Description	1	Resets the Digital Comparator 4 trigger unit to its initial conditions.	0	No effect.
Value	Description									
1	Resets the Digital Comparator 4 trigger unit to its initial conditions.									
0	No effect.									
19	DCTRIG3	R/W	0	<p>Digital Comparator Trigger 3</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Resets the Digital Comparator 3 trigger unit to its initial conditions.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table> <p>When the trigger has been cleared, this bit is automatically cleared. 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.</p>	Value	Description	1	Resets the Digital Comparator 3 trigger unit to its initial conditions.	0	No effect.
Value	Description									
1	Resets the Digital Comparator 3 trigger unit to its initial conditions.									
0	No effect.									
18	DCTRIG2	R/W	0	<p>Digital Comparator Trigger 2</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Resets the Digital Comparator 2 trigger unit to its initial conditions.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table> <p>When the trigger has been cleared, this bit is automatically cleared. 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.</p>	Value	Description	1	Resets the Digital Comparator 2 trigger unit to its initial conditions.	0	No effect.
Value	Description									
1	Resets the Digital Comparator 2 trigger unit to its initial conditions.									
0	No effect.									

Bit/Field	Name	Type	Reset	Description
17	DCTRIG1	R/W	0	<p>Digital Comparator Trigger 1</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 1 trigger unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the trigger has been cleared, this bit is automatically cleared. 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.</p>
16	DCTRIG0	R/W	0	<p>Digital Comparator Trigger 0</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 0 trigger unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the trigger has been cleared, this bit is automatically cleared. 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.</p>
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	<p>Digital Comparator Interrupt 7</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 7 interrupt unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the interrupt has been cleared, this bit is automatically cleared. 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>
6	DCINT6	R/W	0	<p>Digital Comparator Interrupt 6</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 6 interrupt unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the interrupt has been cleared, this bit is automatically cleared. 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
5	DCINT5	R/W	0	<p>Digital Comparator Interrupt 5</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 5 interrupt unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the interrupt has been cleared, this bit is automatically cleared. 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 Interrupt 4</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 4 interrupt unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the interrupt has been cleared, this bit is automatically cleared. 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 Interrupt 3</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 3 interrupt unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the interrupt has been cleared, this bit is automatically cleared. 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 Interrupt 2</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 2 interrupt unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the interrupt has been cleared, this bit is automatically cleared. 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
1	DCINT1	R/W	0	<p>Digital Comparator Interrupt 1</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 1 interrupt unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the interrupt has been cleared, this bit is automatically cleared. 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>
0	DCINT0	R/W	0	<p>Digital Comparator Interrupt 0</p> <p>Value Description</p> <p>1 Resets the Digital Comparator 0 interrupt unit to its initial conditions.</p> <p>0 No effect.</p> <p>When the interrupt has been cleared, this bit is automatically cleared. 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 39: ADC Digital Comparator Control 0 (ADCDCCTL0), offset 0xE00**
- Register 40: ADC Digital Comparator Control 1 (ADCDCCTL1), offset 0xE04**
- Register 41: ADC Digital Comparator Control 2 (ADCDCCTL2), offset 0xE08**
- Register 42: ADC Digital Comparator Control 3 (ADCDCCTL3), offset 0xE0C**
- Register 43: ADC Digital Comparator Control 4 (ADCDCCTL4), offset 0xE10**
- Register 44: ADC Digital Comparator Control 5 (ADCDCCTL5), offset 0xE14**
- Register 45: ADC Digital Comparator Control 6 (ADCDCCTL6), offset 0xE18**
- Register 46: 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
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved				CTE	CTC	CTM	reserved				CIE	CIC	CIM		
Type	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	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

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	Comparison Trigger Enable
	Value	Description		
	1	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.		
	0	Disables the trigger function state machine. ADC conversion data is ignored by the trigger function.		

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 COMP0 and COMP1. The COMP0 and COMP1 fields are defined in the ADCDCCMPx registers.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Low Band ADC Data < COMP0 and < COMP1</td></tr> <tr> <td>0x1</td><td>Mid Band COMP0 ≤ ADC Data < COMP1</td></tr> <tr> <td>0x2</td><td>reserved</td></tr> <tr> <td>0x3</td><td>High Band COMP0 ≤ COMP1 ≤ ADC Data</td></tr> </tbody> </table>	Value	Description	0x0	Low Band ADC Data < COMP0 and < COMP1	0x1	Mid Band COMP0 ≤ ADC Data < COMP1	0x2	reserved	0x3	High Band COMP0 ≤ COMP1 ≤ ADC Data
Value	Description													
0x0	Low Band ADC Data < COMP0 and < COMP1													
0x1	Mid Band COMP0 ≤ ADC Data < COMP1													
0x2	reserved													
0x3	High Band COMP0 ≤ COMP1 ≤ ADC Data													
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> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Always This mode generates a trigger every time the ADC conversion data falls within the selected operational region.</td></tr> <tr> <td>0x1</td><td>Once This mode generates a trigger the first time that the ADC conversion data enters the selected operational region.</td></tr> <tr> <td>0x2</td><td>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 CTC encodings of 0x0 and 0x3.</td></tr> <tr> <td>0x3</td><td>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 CTC encodings of 0x0 and 0x3.</td></tr> </tbody> </table>	Value	Description	0x0	Always This mode generates a trigger every time the ADC conversion data falls within the selected operational region.	0x1	Once This mode generates a trigger the first time that the ADC conversion data enters the selected operational region.	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 CTC encodings of 0x0 and 0x3.	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 CTC encodings of 0x0 and 0x3.
Value	Description													
0x0	Always This mode generates a trigger every time the ADC conversion data falls within the selected operational region.													
0x1	Once This mode generates a trigger the first time that the ADC conversion data enters the selected operational region.													
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 CTC encodings of 0x0 and 0x3.													
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 CTC encodings of 0x0 and 0x3.													
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	CIE	R/W	0	<p>Comparison Interrupt Enable</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Enables the comparison interrupt. The ADC conversion data is used to determine if an interrupt should be generated according to the programming of the CIC and CIM fields.</td></tr> <tr> <td>0</td><td>Disables the comparison interrupt. ADC conversion data has no effect on interrupt generation.</td></tr> </tbody> </table>	Value	Description	1	Enables the comparison interrupt. The ADC conversion data is used to determine if an interrupt should be generated according to the programming of the CIC and CIM fields.	0	Disables the comparison interrupt. ADC conversion data has no effect on interrupt generation.				
Value	Description													
1	Enables the comparison interrupt. The ADC conversion data is used to determine if an interrupt should be generated according to the programming of the CIC and CIM fields.													
0	Disables the comparison interrupt. ADC conversion data has no effect on interrupt generation.													

Bit/Field	Name	Type	Reset	Description										
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 COMP0 and COMP1. The COMP0 and COMP1 fields are defined in the ADCDCMPx registers.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Low Band ADC Data < COMP0 and < COMP1</td></tr> <tr> <td>0x1</td><td>Mid Band COMP0 ≤ ADC Data < COMP1</td></tr> <tr> <td>0x2</td><td>reserved</td></tr> <tr> <td>0x3</td><td>High Band COMP0 < COMP1 ≤ ADC Data</td></tr> </tbody> </table>	Value	Description	0x0	Low Band ADC Data < COMP0 and < COMP1	0x1	Mid Band COMP0 ≤ ADC Data < COMP1	0x2	reserved	0x3	High Band COMP0 < COMP1 ≤ ADC Data
Value	Description													
0x0	Low Band ADC Data < COMP0 and < COMP1													
0x1	Mid Band COMP0 ≤ ADC Data < COMP1													
0x2	reserved													
0x3	High Band COMP0 < COMP1 ≤ ADC Data													
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> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Always This mode generates an interrupt every time the ADC conversion data falls within the selected operational region.</td></tr> <tr> <td>0x1</td><td>Once This mode generates an interrupt the first time that the ADC conversion data enters the selected operational region.</td></tr> <tr> <td>0x2</td><td>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.</td></tr> <tr> <td>0x3</td><td>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.</td></tr> </tbody> </table>	Value	Description	0x0	Always This mode generates an interrupt every time the ADC conversion data falls within the selected operational region.	0x1	Once This mode generates an interrupt the first time that the ADC conversion data enters the selected operational region.	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.	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.
Value	Description													
0x0	Always This mode generates an interrupt every time the ADC conversion data falls within the selected operational region.													
0x1	Once This mode generates an interrupt the first time that the ADC conversion data enters the selected operational region.													
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.													
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.													

Register 47: ADC Digital Comparator Range 0 (ADCDCCMP0), offset 0xE40**Register 48: ADC Digital Comparator Range 1 (ADCDCCMP1), offset 0xE44****Register 49: ADC Digital Comparator Range 2 (ADCDCCMP2), offset 0xE48****Register 50: ADC Digital Comparator Range 3 (ADCDCCMP3), offset 0xE4C****Register 51: ADC Digital Comparator Range 4 (ADCDCCMP4), offset 0xE50****Register 52: ADC Digital Comparator Range 5 (ADCDCCMP5), offset 0xE54****Register 53: ADC Digital Comparator Range 6 (ADCDCCMP6), offset 0xE58****Register 54: 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: The value in the COMP1 field must be greater than or equal to the value in the COMP0 field or unexpected results can occur.

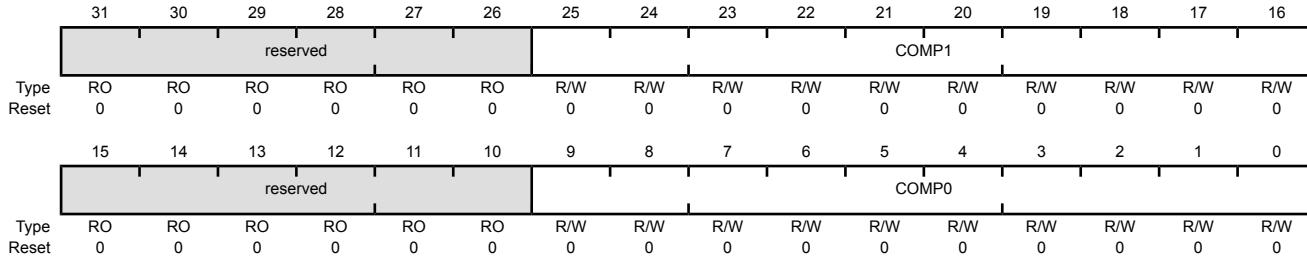
ADC Digital Comparator Range 0 (ADCDCCMP0)

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.

13 Universal Asynchronous Receivers/Transmitters (UARTs)

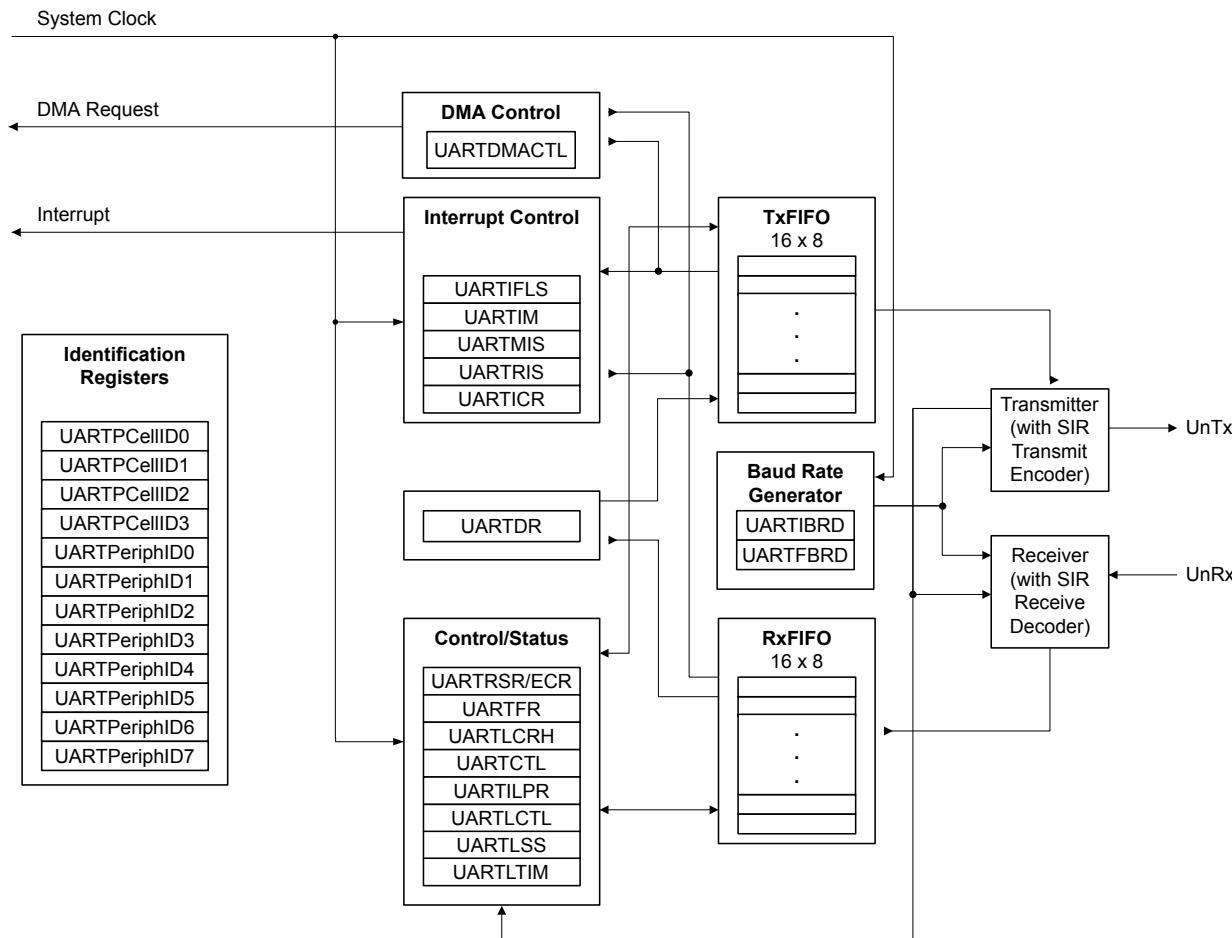
The Stellaris® LM3S9L97 controller includes three Universal Asynchronous Receiver/Transmitter (UART) with the following features:

- 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
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
- Standard asynchronous communication bits for start, stop, and parity
- Line-break generation and detection
- Fully programmable serial interface characteristics
 - 5, 6, 7, or 8 data bits
 - Even, odd, stick, or no-parity bit generation/detection
 - 1 or 2 stop bit generation
- 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 μ 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)
- LIN protocol support
- Standard FIFO-level and End-of-Transmission interrupts
- Efficient transfers using Micro Direct Memory Access Controller (μ 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

13.1 Block Diagram

Figure 13-1. UART Module Block Diagram



13.2 Signal Description

Table 13-1 on page 629 and Table 13-2 on page 629 list the external signals of the UART module and describe the function of each. The UART signals are alternate functions for some GPIO signals and default to be GPIO signals at reset, with the exception of the U0Rx and U0Tx pins which default to the UART function. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these UART signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 447) should be set to choose the UART function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 465) to assign the UART signal to the specified GPIO port pin. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 423.

Table 13-1. Signals for UART (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
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	PE6 (9) PD0 (9) PA6 (9)	I	TTL	UART module 1 Clear To Send modem status input signal.
U1DCD	1 11 35	PE7 (9) PD1 (9) PA7 (9)	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
U1DSR	47	PF0 (9)	I	TTL	UART module 1 Data Set Ready modem output control line.
U1DTR	100	PD7 (9)	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
U1RI	97	PD4 (9)	I	TTL	UART module 1 Ring Indicator modem status input signal.
U1RTS	61	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. When in IrDA mode, this signal has IrDA modulation.
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. When in IrDA mode, this signal has IrDA modulation.
U2Rx	10 19 92 98	PD0 (4) PG0 (1) PB4 (4) PD5 (9)	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
U2Tx	6 11 18 99	PE4 (5) PD1 (4) PG1 (1) PD6 (9)	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.

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

Table 13-2. Signals for UART (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
U0Rx	L3	PA0 (1)	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
U0Tx	M3	PA1 (1)	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
U1CTS	A1 G1 L6	PE6 (9) PD0 (9) PA6 (9)	I	TTL	UART module 1 Clear To Send modem status input signal.

Table 13-2. Signals for UART (108BGA) (continued)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
U1DCD	B1 G2 M6	PE7 (9) PD1 (9) PA7 (9)	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
U1DSR	M9	PF0 (9)	I	TTL	UART module 1 Data Set Ready modem output control line.
U1DTR	A2	PD7 (9)	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
U1RI	B5	PD4 (9)	I	TTL	UART module 1 Ring Indicator modem status input signal.
U1RTS	H12	PF1 (9)	O	TTL	UART module 1 Request to Send modem output control line.
U1Rx	G1 H2 M2 L3 E12 A6	PD0 (5) PD2 (1) PC6 (5) PA0 (9) PB0 (5) PB4 (7)	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
U1Tx	G2 H1 L2 M3 D12 B7	PD1 (5) PD3 (1) PC7 (5) PA1 (9) PB1 (5) PB5 (7)	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
U2Rx	G1 K1 A6 C6	PD0 (4) PG0 (1) PB4 (4) PD5 (9)	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
U2Tx	B2 G2 K2 A3	PE4 (5) PD1 (4) PG1 (1) PD6 (9)	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.

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

13.3 Functional Description

Each Stellaris UART performs the functions of parallel-to-serial and serial-to-parallel conversions. It is similar in functionality to a 16C550 UART, but is not register compatible.

The UART is configured for transmit and/or receive via the TXE and RXE bits of the **UART Control (UARTCTL)** register (see page 654). Transmit and receive are both enabled out of reset. Before any control registers are programmed, the UART must be disabled by clearing the UARTE 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 module also includes a serial IR (SIR) encoder/decoder block that can be connected to an infrared transceiver to implement an IrDA SIR physical layer. The SIR function is programmed using the **UARTCTL** register.

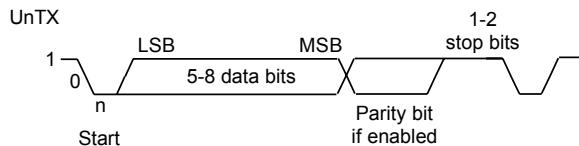
13.3.1 Transmit/Receive Logic

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. The control logic outputs the serial bit stream beginning with a start bit and followed by the data bits

(LSB first), parity bit, and the stop bits according to the programmed configuration in the control registers. See Figure 13-2 on page 631 for details.

The receive logic performs serial-to-parallel conversion on the received bit stream after a valid start pulse has been detected. Overrun, parity, frame error checking, and line-break detection are also performed, and their status accompanies the data that is written to the receive FIFO.

Figure 13-2. UART Character Frame



13.3.2 Baud-Rate Generation

The baud-rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. The number formed by these two values is used by the baud-rate generator to determine the bit period. Having a fractional baud-rate divider allows the UART to generate all the standard baud rates.

The 16-bit integer is loaded through the **UART Integer Baud-Rate Divisor (UARTIBRD)** register (see page 650) and the 6-bit fractional part is loaded with the **UART Fractional Baud-Rate Divisor (UARTFBRD)** register (see page 651). The baud-rate divisor (BRD) has the following relationship to the system clock (where *BRDI* is the integer part of the BRD and *BRDF* is the fractional part, separated by a decimal place.)

$$\text{BRD} = \text{BRDI} + \text{BRDF} = \text{UARTSysClk} / (\text{ClkDiv} * \text{Baud Rate})$$

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 on 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 652), the **UARTIBRD** and **UARTFBRD** registers form an internal 30-bit register. This internal register is only updated when a write operation to **UARTLCRH** is performed, so any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register for the changes to take effect.

To update the baud-rate registers, there are four possible sequences:

- **UARTIBRD** write, **UARTFBRD** write, and **UARTLCRH** write
- **UARTFBRD** write, **UARTIBRD** write, and **UARTLCRH** write
- **UARTIBRD** write and **UARTLCRH** write
- **UARTFBRD** write and **UARTLCRH** write

13.3.3 Data Transmission

Data received or transmitted is stored in two 16-byte FIFOs, though the receive FIFO has an extra four bits per character for status information. For transmission, data is written into the transmit FIFO. If the UART is enabled, it causes a data frame to start transmitting with the parameters indicated in the **UARTLCRH** register. Data continues to be transmitted until there is no data left in the transmit FIFO. The **BUSY** bit in the **UART Flag (UARTFR)** register (see page 646) 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` signal 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 on the setting of the `HSE` bit (bit 5) in **UARTCTL** (described in “Transmit/Receive Logic” on page 630).

The start bit is valid and recognized if the `UnRx` signal is still low on the eighth cycle of Baud16 (`HSE` clear) or the fourth cycle of Baud 8 (`HSE` set), otherwise it is ignored. After a valid start bit is detected, 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 is enabled. Data length and parity are defined in the **UARTLCRH** register.

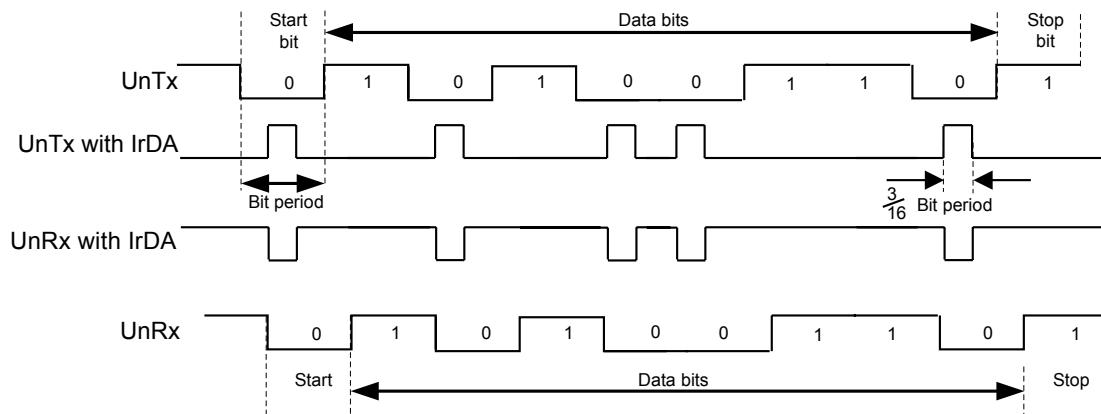
Lastly, a valid stop bit is confirmed if the `UnRx` signal is High, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO along with any error bits associated with that word.

13.3.4 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 a 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. When enabled, the SIR block uses the `UnTx` and `UnRx` pins for the SIR protocol. These signals should be connected to an infrared transceiver to implement an IrDA SIR physical layer link. 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. The SIR block has two modes of operation:

- In normal IrDA mode, a zero logic level is transmitted as a 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 and driving 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 μ s, assuming a nominal 1.8432 MHz frequency) by changing the appropriate bit in the **UARTCR** register. See page 649 for more information on IrDA low-power pulse-duration configuration.

Figure 13-3 on page 633 shows the UART transmit and receive signals, with and without IrDA modulation.

Figure 13-3. IrDA Data Modulation

In both normal and low-power IrDA modes:

- During transmission, the UART data bit is used as the base for encoding
- During reception, the decoded bits are transferred to the UART receive logic

The IrDA SIR physical layer specifies a half-duplex communication link, with a minimum 10-ms delay between transmission and reception. This delay must be generated by software because it is not automatically supported by the UART. The delay is required because the infrared receiver electronics might become biased or even saturated from the optical power coupled from the adjacent transmitter LED. This delay is known as latency or receiver setup time.

13.3.5 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** signal is used as a bit clock, and the **UnRx** signal is used as the half-duplex communication line connected to the smartcard. A GPIO signal can be used to generate the reset signal to the smartcard. The remaining smartcard signals should be provided by the system design.

When using ISO 7816 mode, the **UARTLCRH** register must be set to transmit 8-bit words (**WLEN** bits 6:5 configured to 0x3) with EVEN parity (**PEN** set and **EPS** set). In this mode, the UART automatically uses 2 stop bits, and the **STP2** bit of the **UARTLCRH** register is ignored.

If a parity error is detected during transmission, **UnRx** is pulled Low during the second stop bit. In this case, the UART aborts the transmission, flushes the transmit FIFO and discards any data it contains, and raises 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.

13.3.6 Modem Handshake Support

This section describes how to configure and use the modem status signals for UART1 when connected as a DTE (data terminal equipment) or as a DCE (data communications equipment). In general, a modem is a DCE and a computing device that connects to a modem is the DTE.

13.3.6.1 Signaling

The status signals provided by UART1 differ based on whether the UART is used as a DTE or DCE. When used as a DTE, the modem status signals are defined as:

- $\overline{U1CTS}$ is Clear To Send
- $\overline{U1DSR}$ is Data Set Ready
- $\overline{U1DCD}$ is Data Carrier Detect
- $\overline{U1RI}$ is Ring Indicator
- $\overline{U1RTS}$ is Request To Send
- $\overline{U1DTR}$ is Data Terminal Ready

When used as a DCE, the the modem status signals are defined as:

- $\overline{U1CTS}$ is Request To Send
- $\overline{U1DSR}$ is Data Terminal Ready
- $\overline{U1RTS}$ is Clear To Send
- $\overline{U1DTR}$ is Data Set Ready

Note that the support for DCE functions Data Carrier Detect and Ring Indicator are not provided. If these signals are required, their function can be emulated by using a general-purpose I/O signal and providing software support.

13.3.6.2 Flow Control Methods

Flow control can be accomplished by either hardware or software. The following sections describe the different methods.

Hardware Flow Control (RTS/CTS)

Hardware flow control between two devices is accomplished by connecting the $\overline{U1RTS}$ output to the Clear-To-Send input on the receiving device, and connecting the Request-To-Send output on the receiving device to the $\overline{U1CTS}$ input.

The $\overline{U1CTS}$ input controls the transmitter. The transmitter may only transmit data when the $\overline{U1CTS}$ input is asserted. The $\overline{U1RTS}$ output signal indicates the state of the receive FIFO. $\overline{U1CTS}$ remains asserted until the preprogrammed watermark level is reached, indicating that the Receive FIFO has no space to store additional characters.

The **UARTCTL** register bits 15 (CTSEN) and 14 (RTSEN) specify the flow control mode as shown in Table 13-3 on page 634.

Table 13-3. Flow Control Mode

CTSEN	RTSEN	Description
1	1	RTS and CTS flow control enabled
1	0	Only CTS flow control enabled
0	1	Only RTS flow control enabled
0	0	Both RTS and CTS flow control disabled

Note that when RTSEN is 1, software cannot modify the $\overline{U1RTS}$ output value through the **UARTCTL** register Request to Send (RTS) bit, and the status of the RTS bit should be ignored.

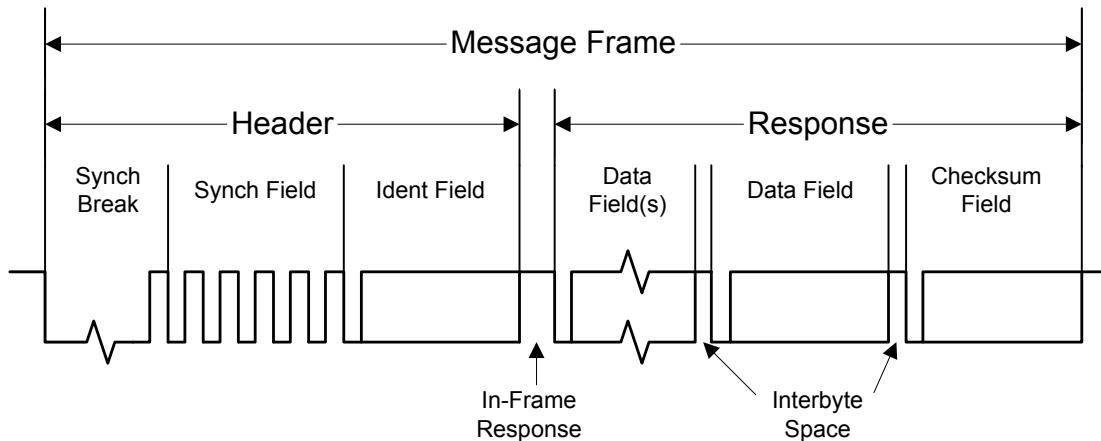
Software Flow Control (Modem Status Interrupts)

Software flow control between two devices is accomplished by using interrupts to indicate the status of the UART. Interrupts may be generated for **U1DSR**, **U1DCD**, **U1CTS**, and **U1RI** using the **UARTIM** bits 3 through 0 respectively. The raw and masked interrupt status may be checked using the **UARTRIS** and **UARTMIS** register. These interrupts may be cleared using the **UARTICR** register.

13.3.7 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). Figure 13-4 on page 635 illustrates the structure of a LIN message.

Figure 13-4. LIN Message



The UART should be configured as follows 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 and the Identifier data at location 1, followed by the data to be transmitted, and with the checksum in the final FIFO entry.

13.3.7.1 LIN Master

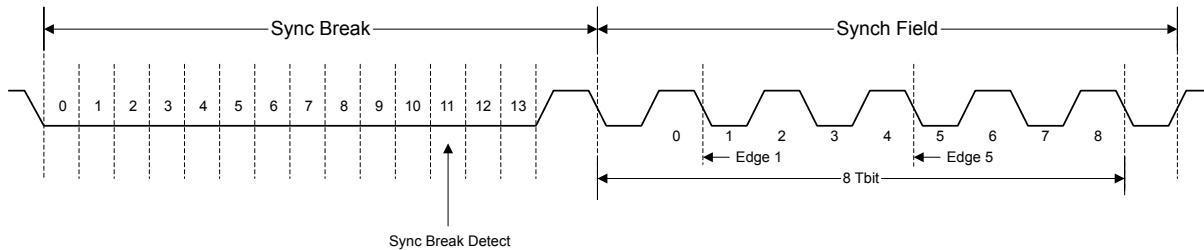
The UART is enabled to be the LIN master by setting the **MASTER** bit in the **UARTLCTL** register. The length of the Sync Break is programmable using the **BLEN** field in the **UARTLCTL** register and can be 13-16 bits (baud clock cycles).

13.3.7.2 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. Figure 13-5 on page 636 illustrates the synchronization field.

Figure 13-5. LIN Synchronization Field



13.3.8 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 641). 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 652).

FIFO status can be monitored via the **UART Flag (UARTFR)** register (see page 646) 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 658). Both FIFOs can be individually configured to trigger interrupts at different levels. Available configurations include $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, $\frac{3}{4}$, and $\frac{7}{8}$. For example, if the $\frac{1}{4}$ option is selected for the receive FIFO, the UART generates a receive interrupt after 4 data bytes are received. Out of reset, both FIFOs are configured to trigger an interrupt at the $\frac{1}{2}$ mark.

13.3.9 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 **UARTCTL** 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 667).

The interrupt events that can trigger a controller-level interrupt are defined in the **UART Interrupt Mask (UARTIM)** register (see page 660) by setting the corresponding IM bits. If interrupts are not used, the raw interrupt status is always visible via the **UART Raw Interrupt Status (UARTRIS)** register (see page 664).

Interrupts are always cleared (for both the **UARTMIS** and **UARTRIS** registers) by writing a 1 to the corresponding bit in the **UART Interrupt Clear (UARTICR)** register (see page 670).

The receive timeout interrupt is asserted when the receive FIFO is not empty, and no further data is received over a 32-bit period. The receive timeout interrupt is cleared either when the FIFO becomes empty through reading all the data (or by reading the holding register), or when a 1 is written to the corresponding bit in the **UARTICR** register.

13.3.10 Loopback Operation

The UART can be placed into an internal loopback mode for diagnostic or debug work by setting the LBE bit in the **UARTCTL** register (see page 654). In loopback mode, data transmitted on the UnTx output is received on the UnRx input.

13.3.11 DMA Operation

The UART provides an interface to the μDMA controller with separate channels for transmit and receive. The DMA operation of the UART is enabled through the **UART DMA Control (UARTDMACTL)** register. When DMA operation is enabled, the UART asserts 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 any data is 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 configured in the **UARTIFLS** register. 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 μDMA controller depending on how the DMA channel is configured.

To enable DMA operation for the receive channel, set the RXDMAE bit of the **DMA Control (UARTDMACTL)** register. To enable DMA operation for the transmit channel, set the TXDMAE bit of the **UARTDMACTL** register. The UART can also be configured to stop using DMA for the receive channel if a receive error occurs. If the DMAERR bit of the **UARTDMACR** register is set and a receive error occurs, the DMA receive requests are automatically disabled. This error condition can be cleared by clearing the appropriate UART error interrupt.

If DMA is enabled, then the μDMA controller triggers an interrupt when a transfer is complete. The interrupt occurs 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 μDMA completion interrupt.

See “Micro Direct Memory Access (μDMA)” on page 365 for more details about programming the μDMA controller.

13.4 Initialization and Configuration

To enable and initialize the UART, the following steps are necessary:

1. The peripheral clock must be enabled by setting the **UART0**, **UART1**, or **UART2** bits in the **RCGC1** register (see page 276).
2. The clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module (see page 285).
3. Set the GPIO **AFSEL** bits for the appropriate pins (see page 447). To determine which GPIOs to configure, see Table 24-4 on page 1182.
4. Configure the GPIO current level and/or slew rate as specified for the mode selected (see page 449 and page 457).
5. Configure the **PMCh** fields in the **GPIOPCTL** register to assign the UART signals to the appropriate pins (see page 465 and Table 24-5 on page 1190).

To use the UARTs, the peripheral clock must be enabled by setting the **UART0**, **UART1**, or **UART2** bits in the **RCGC1** register (see page 276). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module (see page 285). To find out which GPIO port to enable, refer to Table 24-5 on page 1190.

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), because the **UARTIBRD** and **UARTFBRD** registers must be written before the **UARTLCRH** register. Using the equation described in “Baud-Rate Generation” on page 631, 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 650) should be set to 10 decimal or 0xA. The value to be loaded into the **UARTFBRD** register (see page 651) 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 µDMA channel (see “Micro Direct Memory Access (µDMA)” on page 365) and enable the DMA option(s) in the **UARTDMACTL** register.
6. Enable the UART by setting the **UARTEN** bit in the **UARTCTL** register.

13.5 Register Map

Table 13-4 on page 639 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 276).

Note: The UART must be disabled (see the **UARTEN** bit in the **UARTCTL** register on page 654) before any of the control registers are reprogrammed. When the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

Table 13-4. UART Register Map

Offset	Name	Type	Reset	Description	See page
0x000	UARTDR	R/W	0x0000.0000	UART Data	641
0x004	UARTRSR/UARTECR	R/W	0x0000.0000	UART Receive Status/Error Clear	643
0x018	UARTFR	RO	0x0000.0090	UART Flag	646
0x020	UARTILPR	R/W	0x0000.0000	UART IrDA Low-Power Register	649
0x024	UARTIBRD	R/W	0x0000.0000	UART Integer Baud-Rate Divisor	650
0x028	UARTFBRD	R/W	0x0000.0000	UART Fractional Baud-Rate Divisor	651
0x02C	UARTLCRH	R/W	0x0000.0000	UART Line Control	652
0x030	UARTCTL	R/W	0x0000.0300	UART Control	654
0x034	UARTIFLS	R/W	0x0000.0012	UART Interrupt FIFO Level Select	658
0x038	UARTIM	R/W	0x0000.0000	UART Interrupt Mask	660
0x03C	UARTRIS	RO	0x0000.000F	UART Raw Interrupt Status	664
0x040	UARTMIS	RO	0x0000.0000	UART Masked Interrupt Status	667
0x044	UARTICR	W1C	0x0000.0000	UART Interrupt Clear	670
0x048	UARTDMACTL	R/W	0x0000.0000	UART DMA Control	672
0x090	UARTLCTL	R/W	0x0000.0000	UART LIN Control	673
0x094	UARTLSS	RO	0x0000.0000	UART LIN Snap Shot	674

Table 13-4. UART Register Map (*continued*)

Offset	Name	Type	Reset	Description	See page
0x098	UARTLTIM	RO	0x0000.0000	UART LIN Timer	675
0xFD0	UARTPeriphID4	RO	0x0000.0000	UART Peripheral Identification 4	676
0xFD4	UARTPeriphID5	RO	0x0000.0000	UART Peripheral Identification 5	677
0xFD8	UARTPeriphID6	RO	0x0000.0000	UART Peripheral Identification 6	678
0xFDC	UARTPeriphID7	RO	0x0000.0000	UART Peripheral Identification 7	679
0xFE0	UARTPeriphID0	RO	0x0000.0060	UART Peripheral Identification 0	680
0xFE4	UARTPeriphID1	RO	0x0000.0000	UART Peripheral Identification 1	681
0xFE8	UARTPeriphID2	RO	0x0000.0018	UART Peripheral Identification 2	682
0xFEC	UARTPeriphID3	RO	0x0000.0001	UART Peripheral Identification 3	683
0xFF0	UARTPCellID0	RO	0x0000.000D	UART PrimeCell Identification 0	684
0xFF4	UARTPCellID1	RO	0x0000.00F0	UART PrimeCell Identification 1	685
0xFF8	UARTPCellID2	RO	0x0000.0005	UART PrimeCell Identification 2	686
0xFFC	UARTPCellID3	RO	0x0000.00B1	UART PrimeCell Identification 3	687

13.6 Register Descriptions

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

Register 1: UART Data (UARTDR), offset 0x000

Important: Use caution when reading this register. Performing a read may change bit status.

This register is the data register (the interface to the FIFOs).

For transmitted data, if the FIFO is enabled, data written to this location is pushed onto the transmit FIFO. If the FIFO is 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 the FIFO is 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	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.
11	OE	RO	0	UART Overrun Error
	Value	Description		
	1	New data was received when the FIFO was full, resulting in data loss.		
	0	No data has been lost due to a FIFO overrun.		
10	BE	RO	0	UART Break Error
	Value	Description		
	1	A break condition has been 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).		
	0	No break condition has occurred		
	In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the received data input goes to a 1 (marking state), and the next valid start bit is received.			

Bit/Field	Name	Type	Reset	Description
9	PE	RO	0	UART Parity Error Value Description 1 The parity of the received data character does not match the parity defined by bits 2 and 7 of the UARTLCRH register. 0 No parity error has occurred In FIFO mode, this error is associated with the character at the top of the FIFO.
8	FE	RO	0	UART Framing Error Value Description 1 The received character does not have a valid stop bit (a valid stop bit is 1). 0 No framing error has occurred
7:0	DATA	R/W	0x00	Data Transmitted or Received Data that is to be transmitted via the UART is written to this field. When read, this field contains the data that was received by the UART.

Register 2: UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004

The **UARTRSR/UARTECR** register is the receive status register/error clear register.

In addition to the **UARTDR** register, receive status can also be read from the **UARTRSR** register. If the status is read from this register, then the status information corresponds to the entry read from **UARTDR** prior to reading **UARTRSR**. The status information for overrun is set immediately when an overrun condition occurs.

The **UARTRSR** register cannot be written.

A write of any value to the **UARTECR** register clears the framing, parity, break, and overrun errors. All the bits are cleared on reset.

Read-Only Status Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

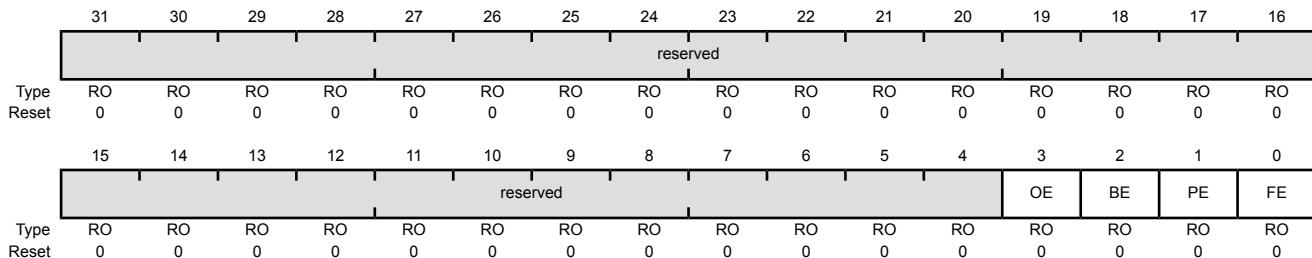
UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0x004

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	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	OE	RO	0	UART Overrun Error

Value Description

- 1 New data was received when the FIFO was full, resulting in data loss.
- 0 No data has been lost due to a FIFO overrun.

This bit is cleared by a write to **UARTECR**.

The FIFO contents remain valid because no further data is written when the FIFO is full, only the contents of the shift register are overwritten. The CPU must read the data in order to empty the FIFO.

Bit/Field	Name	Type	Reset	Description
2	BE	RO	0	UART Break Error
				Value Description
			1	A break condition has been 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).
			0	No break condition has occurred
				This bit is cleared to 0 by a write to UARTECR . 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.
1	PE	RO	0	UART Parity Error
				Value Description
			1	The parity of the received data character does not match the parity defined by bits 2 and 7 of the UARTLCRH register.
			0	No parity error has occurred
				This bit is cleared to 0 by a write to UARTECR .
0	FE	RO	0	UART Framing Error
				Value Description
			1	The received character does not have a valid stop bit (a valid stop bit is 1).
			0	No framing error has occurred
				This bit is cleared to 0 by a write to UARTECR . In FIFO mode, this error is associated with the character at the top of the FIFO.

Write-Only Error Clear Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

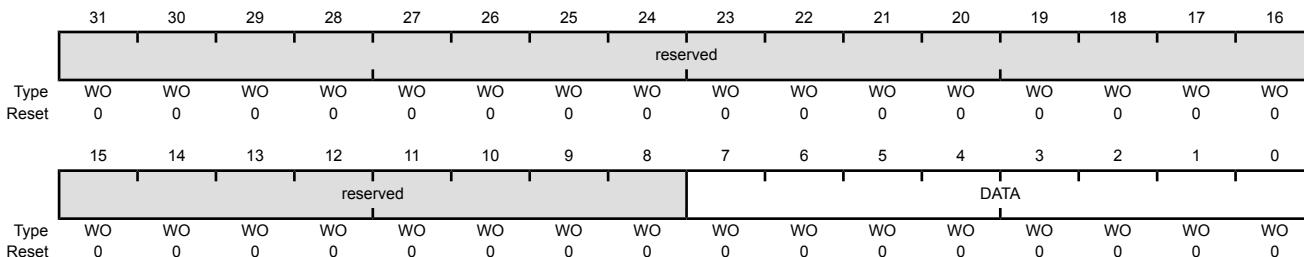
UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0x004

Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	WO	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	WO	0x00	Error Clear A write to this register of any data clears the framing, parity, break, and overrun flags.

Register 3: UART Flag (UARTFR), offset 0x018

The **UARTFR** register is the flag register. After reset, the TXFF, RXFF, and BUSY bits are 0, and TXFE and RXFE bits are 1. The RI, DCD, DSR and CTS bits indicate the modem status.

Note that bits [8,2:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

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	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	RI	RO	0	Ring Indicator Value Description 1 The U1RI signal is asserted. 0 The U1RI signal is not asserted. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
7	TXFE	RO	1	UART Transmit FIFO Empty The meaning of this bit depends on the state of the FEN bit in the UARTLCRH register. Value Description 1 If the FIFO is disabled (FEN is 0), the transmit holding register is empty. If the FIFO is enabled (FEN is 1), the transmit FIFO is empty. 0 The transmitter has data to transmit.

Bit/Field	Name	Type	Reset	Description						
6	RXFF	RO	0	<p>UART Receive FIFO Full The meaning of this bit depends on the state of the FEN bit in the UARTLCRH register.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>If the FIFO is disabled (FEN is 0), the receive holding register is full. If the FIFO is enabled (FEN is 1), the receive FIFO is full.</td></tr> <tr> <td>0</td><td>The receiver can receive data.</td></tr> </tbody> </table>	Value	Description	1	If the FIFO is disabled (FEN is 0), the receive holding register is full. If the FIFO is enabled (FEN is 1), the receive FIFO is full.	0	The receiver can receive data.
Value	Description									
1	If the FIFO is disabled (FEN is 0), the receive holding register is full. If the FIFO is enabled (FEN is 1), the receive FIFO is full.									
0	The receiver can receive data.									
5	TXFF	RO	0	<p>UART Transmit FIFO Full The meaning of this bit depends on the state of the FEN bit in the UARTLCRH register.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>If the FIFO is disabled (FEN is 0), the transmit holding register is full. If the FIFO is enabled (FEN is 1), the transmit FIFO is full.</td></tr> <tr> <td>0</td><td>The transmitter is not full.</td></tr> </tbody> </table>	Value	Description	1	If the FIFO is disabled (FEN is 0), the transmit holding register is full. If the FIFO is enabled (FEN is 1), the transmit FIFO is full.	0	The transmitter is not full.
Value	Description									
1	If the FIFO is disabled (FEN is 0), the transmit holding register is full. If the FIFO is enabled (FEN is 1), the transmit FIFO is full.									
0	The transmitter is not full.									
4	RXFE	RO	1	<p>UART Receive FIFO Empty The meaning of this bit depends on the state of the FEN bit in the UARTLCRH register.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>If the FIFO is disabled (FEN is 0), the receive holding register is empty. If the FIFO is enabled (FEN is 1), the receive FIFO is empty.</td></tr> <tr> <td>0</td><td>The receiver is not empty.</td></tr> </tbody> </table>	Value	Description	1	If the FIFO is disabled (FEN is 0), the receive holding register is empty. If the FIFO is enabled (FEN is 1), the receive FIFO is empty.	0	The receiver is not empty.
Value	Description									
1	If the FIFO is disabled (FEN is 0), the receive holding register is empty. If the FIFO is enabled (FEN is 1), the receive FIFO is empty.									
0	The receiver is not empty.									
3	BUSY	RO	0	<p>UART Busy</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>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.</td></tr> <tr> <td>0</td><td>The UART is not busy.</td></tr> </tbody> </table> <p>This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether UART is enabled).</p>	Value	Description	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.	0	The UART is not busy.
Value	Description									
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.									
0	The UART is not busy.									
2	DCD	RO	0	<p>Data Carrier Detect</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The U1DCD signal is asserted.</td></tr> <tr> <td>0</td><td>The U1DCD signal is not asserted.</td></tr> </tbody> </table> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>	Value	Description	1	The U1DCD signal is asserted.	0	The U1DCD signal is not asserted.
Value	Description									
1	The U1DCD signal is asserted.									
0	The U1DCD signal is not asserted.									

Bit/Field	Name	Type	Reset	Description
1	DSR	RO	0	<p>Data Set Ready</p> <p>Value Description</p> <p>1 The U1DSR signal is asserted.</p> <p>0 The U1DSR signal is not asserted.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>
0	CTS	RO	0	<p>Clear To Send</p> <p>Value Description</p> <p>1 The U1CTS signal is asserted.</p> <p>0 The U1CTS signal is not asserted.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>

Register 4: UART IrDA Low-Power Register (UARTILPR), offset 0x020

The **UARTILPR** register stores the 8-bit 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 when reset.

The internal 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 IrLPBaud16 clock. The low-power divisor value is calculated as follows:

$$\text{ILPDVSR} = \text{SysClk} / F_{\text{IrLPBaud16}}$$

where $F_{\text{IrLPBaud16}}$ is nominally 1.8432 MHz.

The divisor must be programmed such that $1.42 \text{ MHz} < F_{\text{IrLPBaud16}} < 2.12 \text{ MHz}$, resulting in a low-power pulse duration of 1.41–2.11 μs (three times the period of IrLPBaud16). The minimum frequency of IrLPBaud16 ensures that pulses less than one period of IrLPBaud16 are rejected, but pulses greater than 1.4 μs are accepted as valid pulses.

Note: Zero is an illegal value. Programming a zero value results in no IrLPBaud16 pulses being generated.

UART IrDA Low-Power Register (UARTILPR)

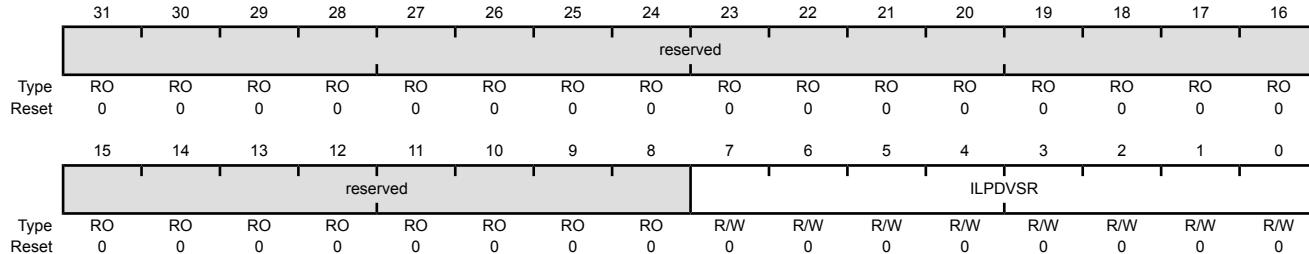
UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0x020

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	ILPDVSR	R/W	0x00	IrDA Low-Power Divisor This field contains the 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 631 for configuration details.

UART Integer Baud-Rate Divisor (UARTIBRD)

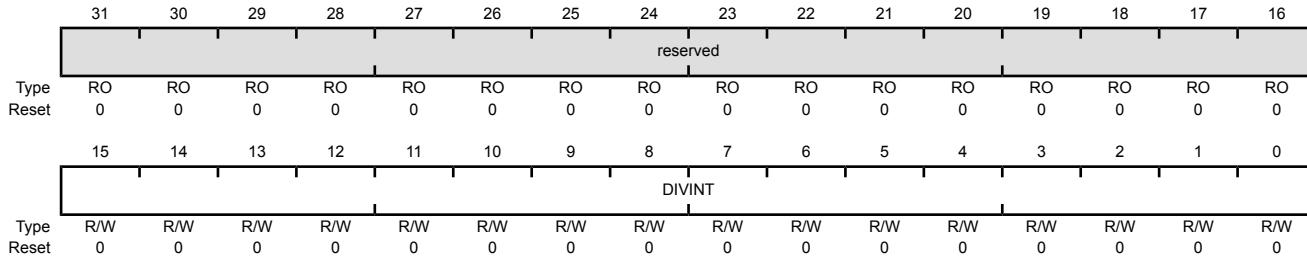
UART0 base: 0x4000.C000

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	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	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 631 for configuration details.

UART Fractional Baud-Rate Divisor (UARTFBRD)

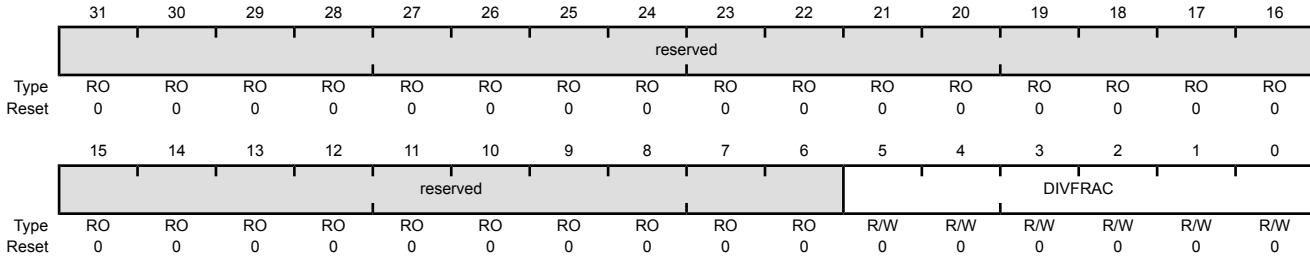
UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0x028

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	DIVFRAC	R/W	0x0	Fractional Baud-Rate Divisor

Register 7: UART Line Control (UARTLCRH), offset 0x02C

The **UARTLCRH** register is the line control register. Serial parameters such as data length, parity, and stop bit selection are implemented in this register.

When updating the baud-rate divisor (**UARTIBRD** and/or **UARTIFRD**), the **UARTLCRH** register must also be written. The write strobe for the baud-rate divisor registers is tied to the **UARTLCRH** register.

UART Line Control (UARTLCRH)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0x02C

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description										
31:8	reserved	RO	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	SPS	R/W	0	UART Stick Parity Select When bits 1, 2, and 7 of UARTLCRH 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	0x0	UART Word Length The bits indicate the number of data bits transmitted or received in a frame as follows: <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>5 bits (default)</td> </tr> <tr> <td>0x1</td> <td>6 bits</td> </tr> <tr> <td>0x2</td> <td>7 bits</td> </tr> <tr> <td>0x3</td> <td>8 bits</td> </tr> </tbody> </table>	Value	Description	0x0	5 bits (default)	0x1	6 bits	0x2	7 bits	0x3	8 bits
Value	Description													
0x0	5 bits (default)													
0x1	6 bits													
0x2	7 bits													
0x3	8 bits													
4	FEN	R/W	0	UART Enable FIFOs <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>The transmit and receive FIFO buffers are enabled (FIFO mode).</td> </tr> <tr> <td>0</td> <td>The FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.</td> </tr> </tbody> </table>	Value	Description	1	The transmit and receive FIFO buffers are enabled (FIFO mode).	0	The FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.				
Value	Description													
1	The transmit and receive FIFO buffers are enabled (FIFO mode).													
0	The FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.													

Bit/Field	Name	Type	Reset	Description
3	STP2	R/W	0	UART Two Stop Bits Select Value Description 1 Two stop bits are transmitted at the end of a frame. The receive logic does not check for two stop bits being received. When in 7816 smartcard mode (the SMART bit is set in the UARTCTL register), the number of stop bits is forced to 2. 0 One stop bit is transmitted at the end of a frame.
2	EPS	R/W	0	UART Even Parity Select Value Description 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. 0 Odd parity is performed, which checks for an odd number of 1s. This bit has no effect when parity is disabled by the PEN bit.
1	PEN	R/W	0	UART Parity Enable Value Description 1 Parity checking and generation is enabled. 0 Parity is disabled and no parity bit is added to the data frame.
0	BRK	R/W	0	UART Send Break Value Description 1 A Low level is continually output on the UnTx signal, after completing transmission of the current character. For the proper execution of the break command, software must set this bit for at least two frames (character periods). 0 Normal use.

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 enable the UART module, the **UARTEN** bit must be set. 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 that bits [15:14,11:10] are only implemented on UART1. These bits are reserved on UART0 and UART2.

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 clearing 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	LIN	HSE	EOT	SMART	SIRLP	SIREN	UARTEN
Type	R/W	R/W	RO	RO	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	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.

Bit/Field	Name	Type	Reset	Description						
15	CTSEN	R/W	0	<p>Enable Clear To Send</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>CTS hardware flow control is enabled. Data is only transmitted when the U1CTS signal is asserted.</td></tr> <tr> <td>0</td><td>CTS hardware flow control is disabled.</td></tr> </tbody> </table> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>	Value	Description	1	CTS hardware flow control is enabled. Data is only transmitted when the U1CTS signal is asserted.	0	CTS hardware flow control is disabled.
Value	Description									
1	CTS hardware flow control is enabled. Data is only transmitted when the U1CTS signal is asserted.									
0	CTS hardware flow control is disabled.									
14	RTSEN	R/W	0	<p>Enable Request to Send</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>RTS hardware flow control is enabled. Data is only requested (by asserting U1RTS) when the receive FIFO has available entries.</td></tr> <tr> <td>0</td><td>RTS hardware flow control is disabled.</td></tr> </tbody> </table> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>	Value	Description	1	RTS hardware flow control is enabled. Data is only requested (by asserting U1RTS) when the receive FIFO has available entries.	0	RTS hardware flow control is disabled.
Value	Description									
1	RTS hardware flow control is enabled. Data is only requested (by asserting U1RTS) when the receive FIFO has available entries.									
0	RTS hardware flow control is disabled.									
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	<p>Request to Send</p> <p>When RTSEN is clear, the status of this bit is reflected on the U1RTS signal. If RTSEN is set, this bit is ignored on a write and should be ignored on read.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>						
10	DTR	R/W	0	<p>Data Terminal Ready</p> <p>This bit sets the state of the U1DTR output.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>						
9	RXE	R/W	1	<p>UART Receive Enable</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The receive section of the UART is enabled.</td></tr> <tr> <td>0</td><td>The receive section of the UART is disabled.</td></tr> </tbody> </table> <p>If the UART is disabled in the middle of a receive, it completes the current character before stopping.</p> <p>Note: To enable reception, the UARLEN bit must also be set.</p>	Value	Description	1	The receive section of the UART is enabled.	0	The receive section of the UART is disabled.
Value	Description									
1	The receive section of the UART is enabled.									
0	The receive section of the UART is disabled.									

Bit/Field	Name	Type	Reset	Description
8	TXE	R/W	1	<p>UART Transmit Enable</p> <p>Value Description</p> <p>1 The transmit section of the UART is enabled.</p> <p>0 The transmit section of the UART is disabled.</p> <p>If the UART is disabled in the middle of a transmission, it completes the current character before stopping.</p> <p>Note: To enable transmission, the UARTEN bit must also be set.</p>
7	LBE	R/W	0	<p>UART Loop Back Enable</p> <p>Value Description</p> <p>1 The UnTx path is fed through the UnRx path.</p> <p>0 Normal operation.</p>
6	LIN	R/W	0	<p>LIN Mode Enable</p> <p>Value Description</p> <p>1 The UART operates in LIN mode.</p> <p>0 Normal operation.</p>
5	HSE	R/W	0	<p>High-Speed Enable</p> <p>Value Description</p> <p>0 The UART is clocked using the system clock divided by 16.</p> <p>1 The UART is clocked using the system clock divided by 8.</p> <p>Note: System clock used is also dependent on the baud-rate divisor configuration (see page 650) and page 651).</p>
4	EOT	R/W	0	<p>End of Transmission</p> <p>This bit determines the behavior of the TXRIS bit in the UARTRIS register.</p> <p>Value Description</p> <p>1 The TXRIS bit is set only after all transmitted data, including stop bits, have cleared the serializer.</p> <p>0 The TXRIS bit is set when the transmit FIFO condition specified in UARTIFLS is met.</p>

Bit/Field	Name	Type	Reset	Description
3	SMART	R/W	0	<p>ISO 7816 Smart Card Support</p> <p>Value Description</p> <p>1 The UART operates in Smart Card mode.</p> <p>0 Normal operation.</p> <p>The application must ensure that it sets 8-bit word length (WLEN set to 0x3) and even parity (PEN set to 1, EPS set to 1, SPS set to 0) in UARTLCRH when using ISO 7816 mode.</p> <p>In this mode, the value of the STP2 bit in UARTLCRH 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.</p> <p>Value Description</p> <p>1 The UART operates in SIR Low-Power mode. Low-level bits are transmitted with a pulse width which is 3 times the period of the IrLPBaud16 input signal, regardless of the selected bit rate.</p> <p>0 Low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period.</p> <p>Setting this bit uses less power, but might reduce transmission distances. See page 649 for more information.</p>
1	SIREN	R/W	0	<p>UART SIR Enable</p> <p>Value Description</p> <p>1 The IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol.</p> <p>0 Normal operation.</p>
0	UARTEN	R/W	0	<p>UART Enable</p> <p>Value Description</p> <p>1 The UART is enabled.</p> <p>0 The UART is disabled.</p> <p>If 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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										RXIFLSEL		TXIFLSEL			
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	1	0	0	1	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.
5:3	RXIFLSEL	R/W	0x2	UART Receive Interrupt FIFO Level Select The trigger points for the receive interrupt are as follows:

Value	Description
0x0	RX FIFO $\geq \frac{1}{8}$ full
0x1	RX FIFO $\geq \frac{1}{4}$ full
0x2	RX FIFO $\geq \frac{1}{2}$ full (default)
0x3	RX FIFO $\geq \frac{3}{4}$ full
0x4	RX FIFO $\geq \frac{7}{8}$ full
0x5-0x7	Reserved

Bit/Field	Name	Type	Reset	Description
2:0	TXIFLSEL	R/W	0x2	UART Transmit Interrupt FIFO Level Select The trigger points for the transmit interrupt are as follows:

Value	Description
0x0	TX FIFO $\leq \frac{1}{8}$ empty
0x1	TX FIFO $\leq \frac{3}{4}$ empty
0x2	TX FIFO $\leq \frac{1}{2}$ empty (default)
0x3	TX FIFO $\leq \frac{1}{4}$ empty
0x4	TX FIFO $\leq \frac{1}{8}$ empty
0x5-0x7	Reserved

Note: If the EOT bit in **UARTCTL** is set (see page 654), 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. Setting a bit allows the corresponding raw interrupt signal to be routed to the interrupt controller. Clearing a bit prevents the raw interrupt signal from being sent to the interrupt controller.

Note that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

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		OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM	DSRIM	DCDIM	CTSIM	RIIM
Type	R/W	R/W	R/W	RO	RO	R/W	R/W	R/W	R/W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	LME5IM	R/W	0	LIN Mode Edge 5 Interrupt Mask
	Value	Description		
	1	An interrupt is sent to the interrupt controller when the LME5RIS bit in the UARTRIS register is set.		
	0	The LME5RIS interrupt is suppressed and not sent to the interrupt controller.		
14	LME1IM	R/W	0	LIN Mode Edge 1 Interrupt Mask
	Value	Description		
	1	An interrupt is sent to the interrupt controller when the LME1RIS bit in the UARTRIS register is set.		
	0	The LME1RIS interrupt is suppressed and not sent to the interrupt controller.		
13	LMSBIM	R/W	0	LIN Mode Sync Break Interrupt Mask
	Value	Description		
	1	An interrupt is sent to the interrupt controller when the LMSBRIS bit in the UARTRIS register is set.		
	0	The LMSBRIS interrupt is suppressed and not sent to the interrupt controller.		

Bit/Field	Name	Type	Reset	Description
12:11	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	OEIM	R/W	0	UART Overrun Error Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the OERIS bit in the UARTRIS register is set. 0 The OERIS interrupt is suppressed and not sent to the interrupt controller.
9	BEIM	R/W	0	UART Break Error Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the BERIS bit in the UARTRIS register is set. 0 The BERIS interrupt is suppressed and not sent to the interrupt controller.
8	PEIM	R/W	0	UART Parity Error Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the PERIS bit in the UARTRIS register is set. 0 The PERIS interrupt is suppressed and not sent to the interrupt controller.
7	FEIM	R/W	0	UART Framing Error Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the FERIS bit in the UARTRIS register is set. 0 The FERIS interrupt is suppressed and not sent to the interrupt controller.
6	RTIM	R/W	0	UART Receive Time-Out Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the RTRIS bit in the UARTRIS register is set. 0 The RTRIS interrupt is suppressed and not sent to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
5	TXIM	R/W	0	UART Transmit Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the TXRIS bit in the UARTRIS register is set. 0 The TXRIS interrupt is suppressed and not sent to the interrupt controller.
4	RXIM	R/W	0	UART Receive Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the RXRIS bit in the UARTRIS register is set. 0 The RXRIS interrupt is suppressed and not sent to the interrupt controller.
3	DSRIM	R/W	0	UART Data Set Ready Modem Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the DSRRIS bit in the UARTRIS register is set. 0 The DSRRIS interrupt is suppressed and not sent to the interrupt controller. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
2	DCDIM	R/W	0	UART Data Carrier Detect Modem Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the DCDRIS bit in the UARTRIS register is set. 0 The DCDRIS interrupt is suppressed and not sent to the interrupt controller. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
1	CTSIM	R/W	0	UART Clear to Send Modem Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the CTSRIS bit in the UARTRIS register is set. 0 The CTSRIS interrupt is suppressed and not sent to the interrupt controller. This bit is implemented only on UART1 and is reserved for UART0 and UART2.

Bit/Field	Name	Type	Reset	Description
0	RIIM	R/W	0	UART Ring Indicator Modem Interrupt Mask
Value Description				
		1		An interrupt is sent to the interrupt controller when the RIRIS bit in the UARTRIS register is set.
		0		The RIRIS interrupt is suppressed and not sent to the interrupt controller.
This bit is implemented only on UART1 and is reserved for UART0 and UART2.				

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.

Note that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

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	DSRRIS	DCCDRIS	CTSRIS	RIRIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	LME5RIS	RO	0	LIN Mode Edge 5 Raw Interrupt Status
	Value Description			
	1	The timer value at the 5th falling edge of the LIN Sync Field has been captured.		
	0	No interrupt		
	This bit is cleared by writing a 1 to the LME5IC bit in the UARTICR register.			
14	LME1RIS	RO	0	LIN Mode Edge 1 Raw Interrupt Status
	Value Description			
	1	The timer value at the 1st falling edge of the LIN Sync Field has been captured.		
	0	No interrupt		
	This bit is cleared by writing a 1 to the LME1IC bit in the UARTICR register.			
13	LMSBRIS	RO	0	LIN Mode Sync Break Raw Interrupt Status
	Value Description			
	1	A LIN Sync Break has been detected.		
	0	No interrupt		
	This bit is cleared by writing a 1 to the LMSBIC bit in the UARTICR register.			

Bit/Field	Name	Type	Reset	Description
12:11	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	OERIS	RO	0	UART Overrun Error Raw Interrupt Status Value Description 1 An overrun error has occurred. 0 No interrupt This bit is cleared by writing a 1 to the OEIC bit in the UARTICR register.
9	BERIS	RO	0	UART Break Error Raw Interrupt Status Value Description 1 A break error has occurred. 0 No interrupt This bit is cleared by writing a 1 to the BEIC bit in the UARTICR register.
8	PERIS	RO	0	UART Parity Error Raw Interrupt Status Value Description 1 A parity error has occurred. 0 No interrupt This bit is cleared by writing a 1 to the PEIC bit in the UARTICR register.
7	FERIS	RO	0	UART Framing Error Raw Interrupt Status Value Description 1 A framing error has occurred. 0 No interrupt This bit is cleared by writing a 1 to the FEIC bit in the UARTICR register.
6	RTRIS	RO	0	UART Receive Time-Out Raw Interrupt Status Value Description 1 A receive time out has occurred. 0 No interrupt This bit is cleared by writing a 1 to the RTIC bit in the UARTICR register.
5	TXRIS	RO	0	UART Transmit Raw Interrupt Status Value Description 1 If the EOT bit in the UARTCTL register is clear, the transmit FIFO level has passed through the condition defined in the UARTIFLS register. If the EOT bit is set, the last bit of all transmitted data and flags has left the serializer. 0 No interrupt This bit is cleared by writing a 1 to the TXIC bit in the UARTICR register.

Bit/Field	Name	Type	Reset	Description						
4	RXRIS	RO	0	<p>UART Receive Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The receive FIFO level has passed through the condition defined in the UARTIFLS register.</td></tr> <tr> <td>0</td><td>No interrupt</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the RXIC bit in the UARTICR register.</p>	Value	Description	1	The receive FIFO level has passed through the condition defined in the UARTIFLS register.	0	No interrupt
Value	Description									
1	The receive FIFO level has passed through the condition defined in the UARTIFLS register.									
0	No interrupt									
3	DSRRIS	RO	0	<p>UART Data Set Ready Modem Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Data Set Ready used for software flow control.</td></tr> <tr> <td>0</td><td>No interrupt</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the DSRIC bit in the UARTICR register.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>	Value	Description	1	Data Set Ready used for software flow control.	0	No interrupt
Value	Description									
1	Data Set Ready used for software flow control.									
0	No interrupt									
2	DCDRIS	RO	0	<p>UART Data Carrier Detect Modem Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Data Carrier Detect used for software flow control.</td></tr> <tr> <td>0</td><td>No interrupt</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the DCDIC bit in the UARTICR register.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>	Value	Description	1	Data Carrier Detect used for software flow control.	0	No interrupt
Value	Description									
1	Data Carrier Detect used for software flow control.									
0	No interrupt									
1	CTSRIS	RO	0	<p>UART Clear to Send Modem Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Clear to Send used for software flow control.</td></tr> <tr> <td>0</td><td>No interrupt</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the CTSIC bit in the UARTICR register.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>	Value	Description	1	Clear to Send used for software flow control.	0	No interrupt
Value	Description									
1	Clear to Send used for software flow control.									
0	No interrupt									
0	RIRIS	RO	0	<p>UART Ring Indicator Modem Raw Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Ring Indicator used for software flow control.</td></tr> <tr> <td>0</td><td>No interrupt</td></tr> </tbody> </table> <p>This bit is cleared by writing a 1 to the RIIC bit in the UARTICR register.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>	Value	Description	1	Ring Indicator used for software flow control.	0	No interrupt
Value	Description									
1	Ring Indicator used for software flow control.									
0	No interrupt									

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.

Note that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

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	DSRMIS	DCCDMIS	CTSMIS	RIMIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	LME5MIS	RO	0	LIN Mode Edge 5 Masked Interrupt Status
		Value	Description	
		1	An unmasked interrupt was signaled due to the 5th falling edge of the LIN Sync Field.	
		0	An interrupt has not occurred or is masked.	
		This bit is cleared by writing a 1 to the LME5IC bit in the UARTICR register.		
14	LME1MIS	RO	0	LIN Mode Edge 1 Masked Interrupt Status
		Value	Description	
		1	An unmasked interrupt was signaled due to the 1st falling edge of the LIN Sync Field.	
		0	An interrupt has not occurred or is masked.	
		This bit is cleared by writing a 1 to the LME1IC bit in the UARTICR register.		
13	LMSBMIS	RO	0	LIN Mode Sync Break Masked Interrupt Status
		Value	Description	
		1	An unmasked interrupt was signaled due to the receipt of a LIN Sync Break.	
		0	An interrupt has not occurred or is masked.	
		This bit is cleared by writing a 1 to the LMSBIC bit in the UARTICR register.		

Bit/Field	Name	Type	Reset	Description
12:11	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	OEMIS	RO	0	UART Overrun Error Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to an overrun error. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the OEIC bit in the UARTICR register.
9	BEMIS	RO	0	UART Break Error Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a break error. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the BEIC bit in the UARTICR register.
8	PEMIS	RO	0	UART Parity Error Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a parity error. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the PEIC bit in the UARTICR register.
7	FEMIS	RO	0	UART Framing Error Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a framing error. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the FEIC bit in the UARTICR register.
6	RTMIS	RO	0	UART Receive Time-Out Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a receive time out. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the RTIC bit in the UARTICR register.
5	TXMIS	RO	0	UART Transmit Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to passing through the specified transmit FIFO level (if the EOT bit is clear) or due to the transmission of the last data bit (if the EOT bit is set). 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the TXIC bit in the UARTICR register.

Bit/Field	Name	Type	Reset	Description
4	RXMIS	RO	0	<p>UART Receive Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to passing through the specified receive FIFO level.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the RXIC bit in the UARTICR register.</p>
3	DSRMIS	RO	0	<p>UART Data Set Ready Modem Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to Data Set Ready.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the DSRIC bit in the UARTICR register.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>
2	DCDMIS	RO	0	<p>UART Data Carrier Detect Modem Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to Data Carrier Detect.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the DCDIC bit in the UARTICR register.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>
1	CTSMIS	RO	0	<p>UART Clear to Send Modem Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to Clear to Send.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the CTSIC bit in the UARTICR register.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>
0	RIMIS	RO	0	<p>UART Ring Indicator Modem Masked Interrupt Status</p> <p>Value Description</p> <p>1 An unmasked interrupt was signaled due to Ring Indicator.</p> <p>0 An interrupt has not occurred or is masked.</p> <p>This bit is cleared by writing a 1 to the RIIC bit in the UARTICR register.</p> <p>This bit is implemented only on UART1 and is reserved for UART0 and UART2.</p>

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.

Note that bits [3:0] are only implemented on UART1. These bits are reserved on UART0 and UART2.

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
	LME5MIC	LME1MIC	LMSBMIC	reserved		OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC	DSRMIC	DCCDMIC	CTSMIC	RIMIC
Type	W1C	W1C	W1C	RO	RO	W1C	W1C	W1C	W1C							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	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	LME5MIC	W1C	0	LIN Mode Edge 5 Interrupt Clear Writing a 1 to this bit clears the LME5RIS bit in the UARTRIS register and the LME5MIS bit in the UARTMIS register.
14	LME1MIC	W1C	0	LIN Mode Edge 1 Interrupt Clear Writing a 1 to this bit clears the LME1RIS bit in the UARTRIS register and the LME1MIS bit in the UARTMIS register.
13	LMSBMIC	W1C	0	LIN Mode Sync Break Interrupt Clear Writing a 1 to this bit clears the LMSBRIS bit in the UARTRIS register and the LMSBMIS bit in the UARTMIS register.
12:11	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	OEIC	W1C	0	Overrun Error Interrupt Clear Writing a 1 to this bit clears the OERIS bit in the UARTRIS register and the OEMIS bit in the UARTMIS register.
9	BEIC	W1C	0	Break Error Interrupt Clear Writing a 1 to this bit clears the BERIS bit in the UARTRIS register and the BEMIS bit in the UARTMIS register.
8	PEIC	W1C	0	Parity Error Interrupt Clear Writing a 1 to this bit clears the PERIS bit in the UARTRIS register and the PEMIS bit in the UARTMIS register.
7	FEIC	W1C	0	Framing Error Interrupt Clear Writing a 1 to this bit clears the FERIS bit in the UARTRIS register and the FEMIS bit in the UARTMIS register.

Bit/Field	Name	Type	Reset	Description
6	RTIC	W1C	0	Receive Time-Out Interrupt Clear Writing a 1 to this bit clears the RTRIS bit in the UARTRIS register and the RTMIS bit in the UARTMIS register.
5	TXIC	W1C	0	Transmit Interrupt Clear Writing a 1 to this bit clears the TXRIS bit in the UARTRIS register and the TXMIS bit in the UARTMIS register.
4	RXIC	W1C	0	Receive Interrupt Clear Writing a 1 to this bit clears the RXRIS bit in the UARTRIS register and the RXMIS bit in the UARTMIS register.
3	DSRMIC	W1C	0	UART Data Set Ready Modem Interrupt Clear Writing a 1 to this bit clears the DSRRIS bit in the UARTRIS register and the DSRMIS bit in the UARTMIS register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
2	DCDMIC	W1C	0	UART Data Carrier Detect Modem Interrupt Clear Writing a 1 to this bit clears the DCDRIS bit in the UARTRIS register and the DCDMIS bit in the UARTMIS register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
1	CTSMIC	W1C	0	UART Clear to Send Modem Interrupt Clear Writing a 1 to this bit clears the CTSRIS bit in the UARTRIS register and the CTSMIS bit in the UARTMIS register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.
0	RIMIC	W1C	0	UART Ring Indicator Modem Interrupt Clear Writing a 1 to this bit clears the RIRIS bit in the UARTRIS register and the RIMIS bit in the UARTMIS register. This bit is implemented only on UART1 and is reserved for UART0 and UART2.

Register 14: UART DMA Control (UARTDMACTL), offset 0x048

The **UARTDMACTL** register is the DMA control register.

UART DMA Control (UARTDMACTL)

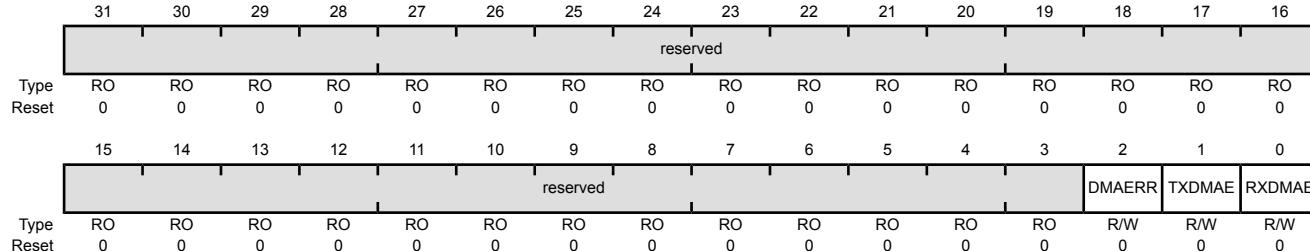
UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0x048

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00000.000	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	DMAERR	R/W	0	DMA on Error
		Value	Description	
		1	μDMA receive requests are automatically disabled when a receive error occurs.	
		0	μDMA receive requests are unaffected when a receive error occurs.	
1	TXDMAE	R/W	0	Transmit DMA Enable
		Value	Description	
		1	μDMA for the transmit FIFO is enabled.	
		0	μDMA for the transmit FIFO is disabled.	
0	RXDMAE	R/W	0	Receive DMA Enable
		Value	Description	
		1	μDMA for the receive FIFO is enabled.	
		0	μDMA for the receive FIFO is disabled.	

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

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

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	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.
5:4	BLEN	R/W	0x0	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 Value Description 1 The UART operates as a LIN master. 0 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)

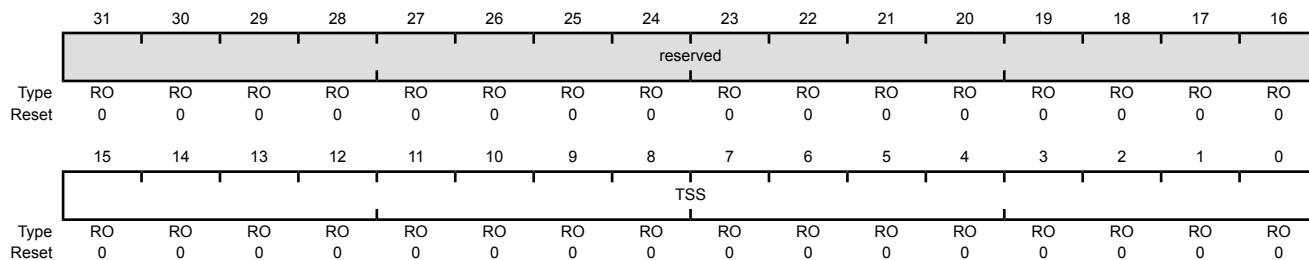
UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0x094

Type RO, reset 0x0000.0000



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	0x0000	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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TIMER															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	TIMER	RO	0x0000	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)

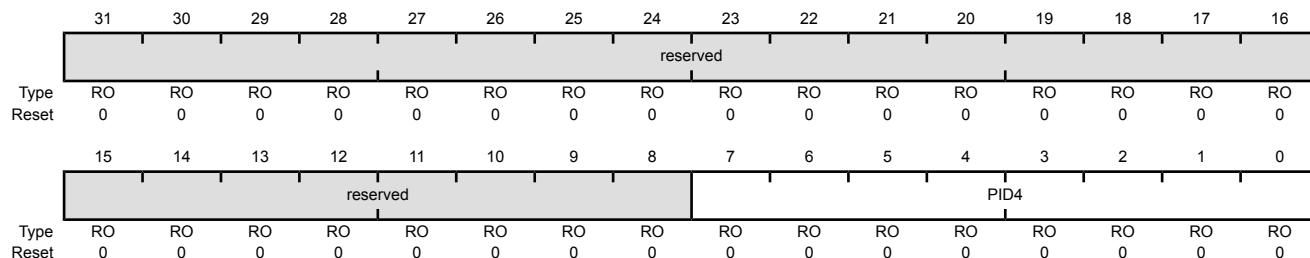
UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFD0

Type RO, reset 0x0000.0000



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)

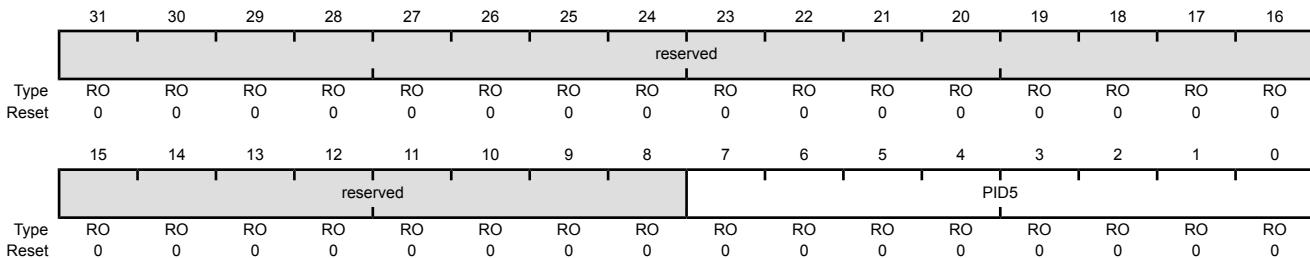
UART0 base: 0x4000.C000

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)

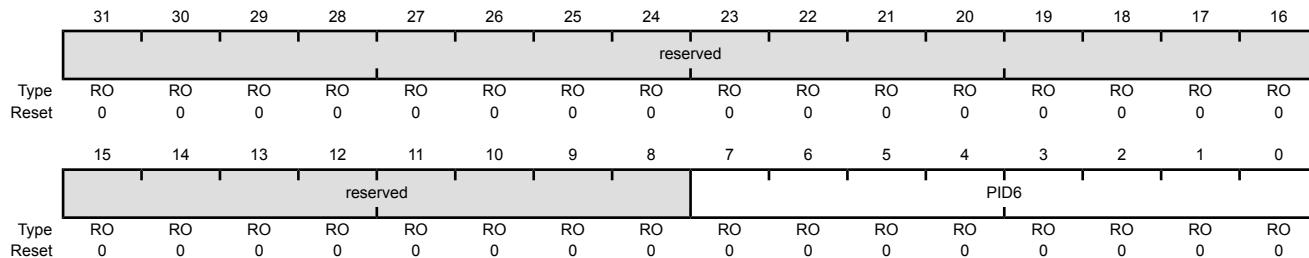
UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFD8

Type RO, reset 0x0000.0000



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)

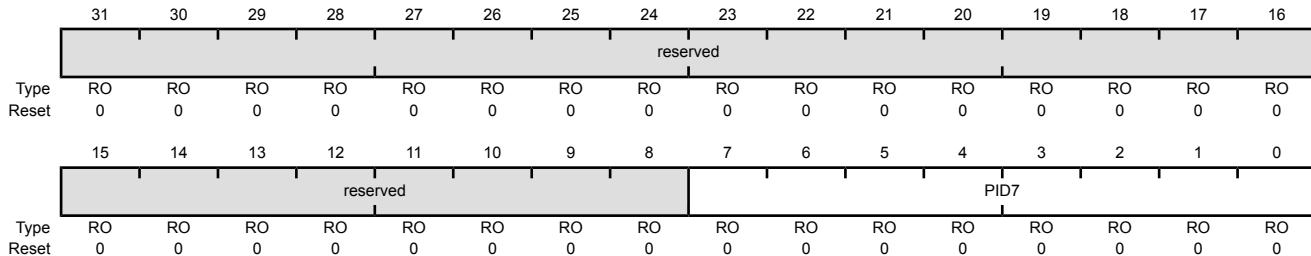
UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFDC

Type RO, reset 0x0000.0000



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)

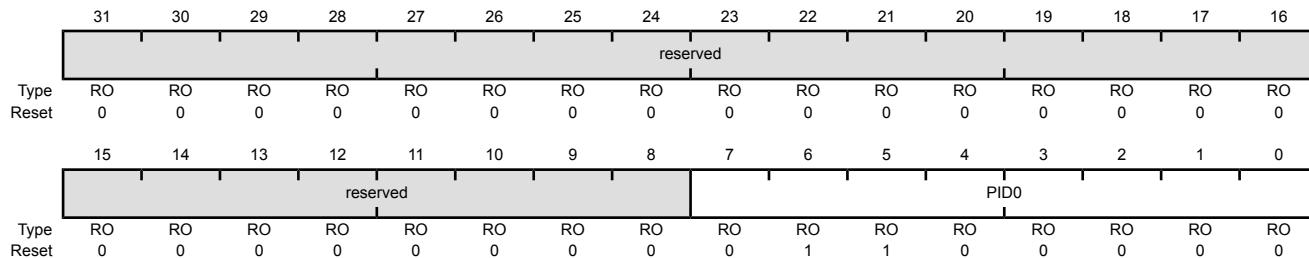
UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFE0

Type RO, reset 0x0000.0060



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)

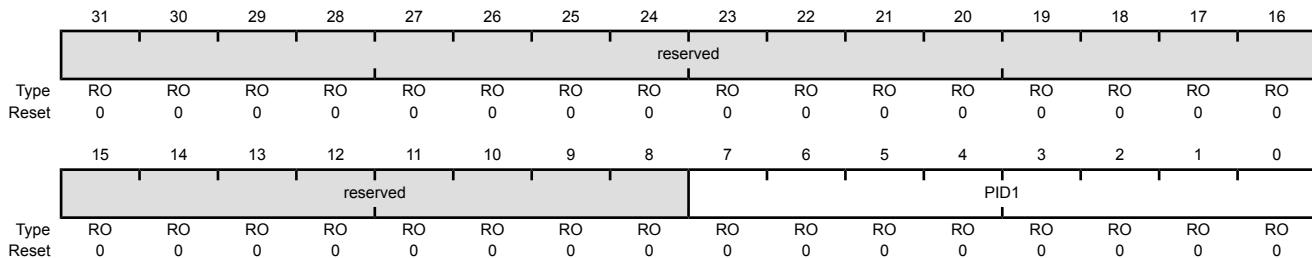
UART0 base: 0x4000.C000

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)

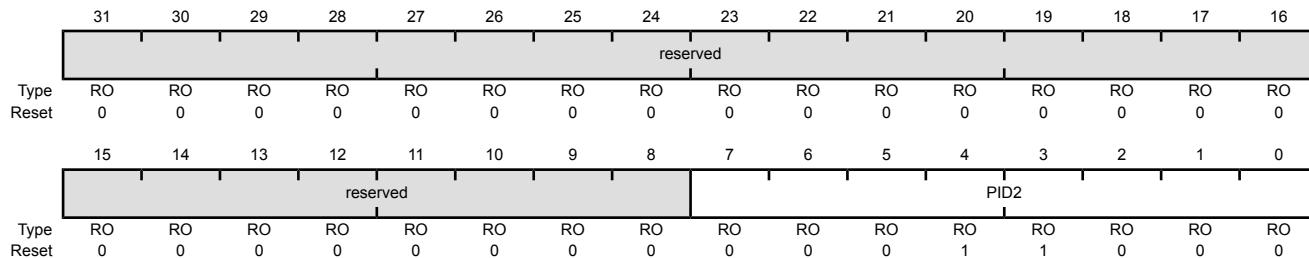
UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFE8

Type RO, reset 0x0000.0018



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)

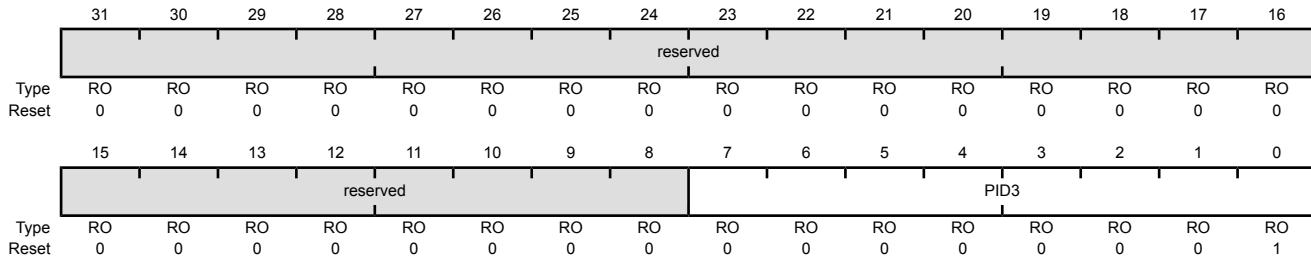
UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFEC

Type RO, reset 0x0000.0001



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)

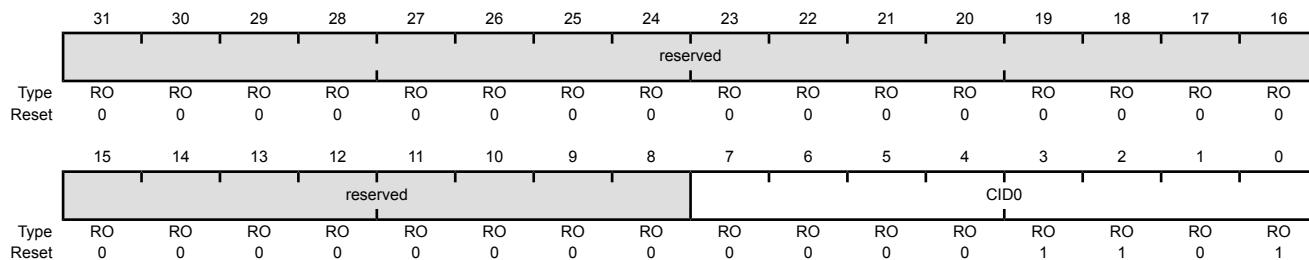
UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFF0

Type RO, reset 0x0000.000D



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)

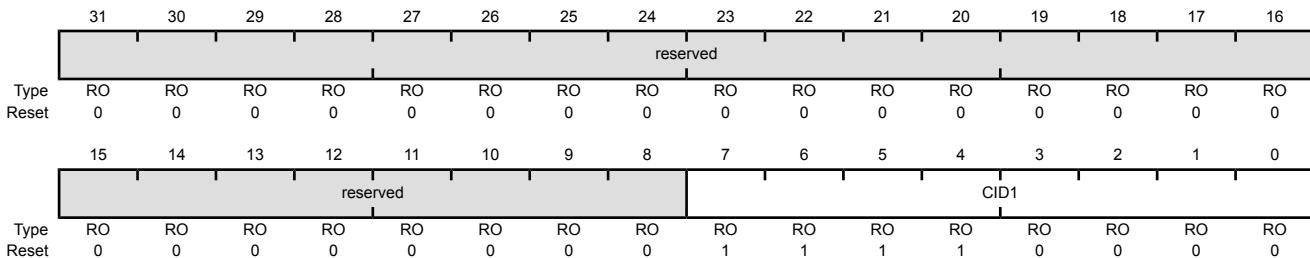
UART0 base: 0x4000.C000

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)

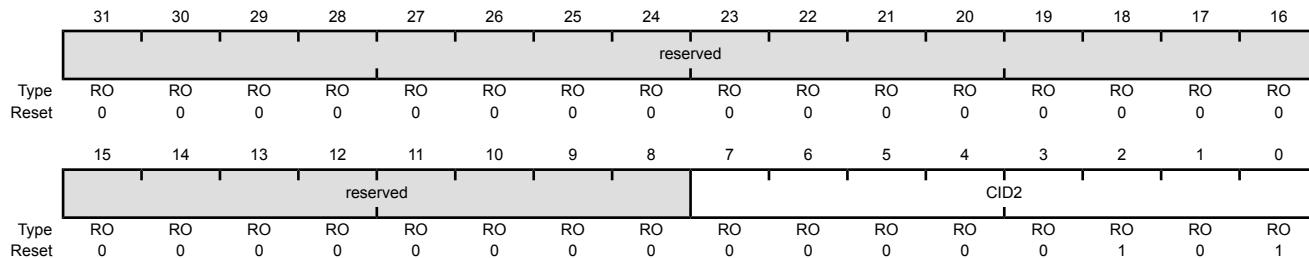
UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFF8

Type RO, reset 0x0000.0005



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)

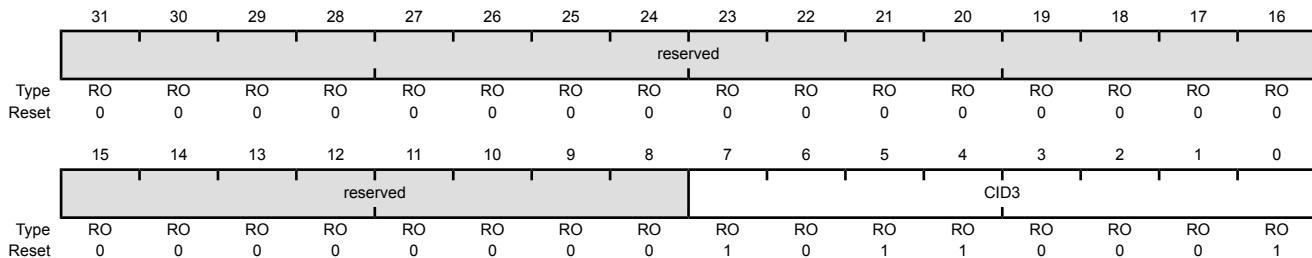
UART0 base: 0x4000.C000

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.

14 Synchronous Serial Interface (SSI)

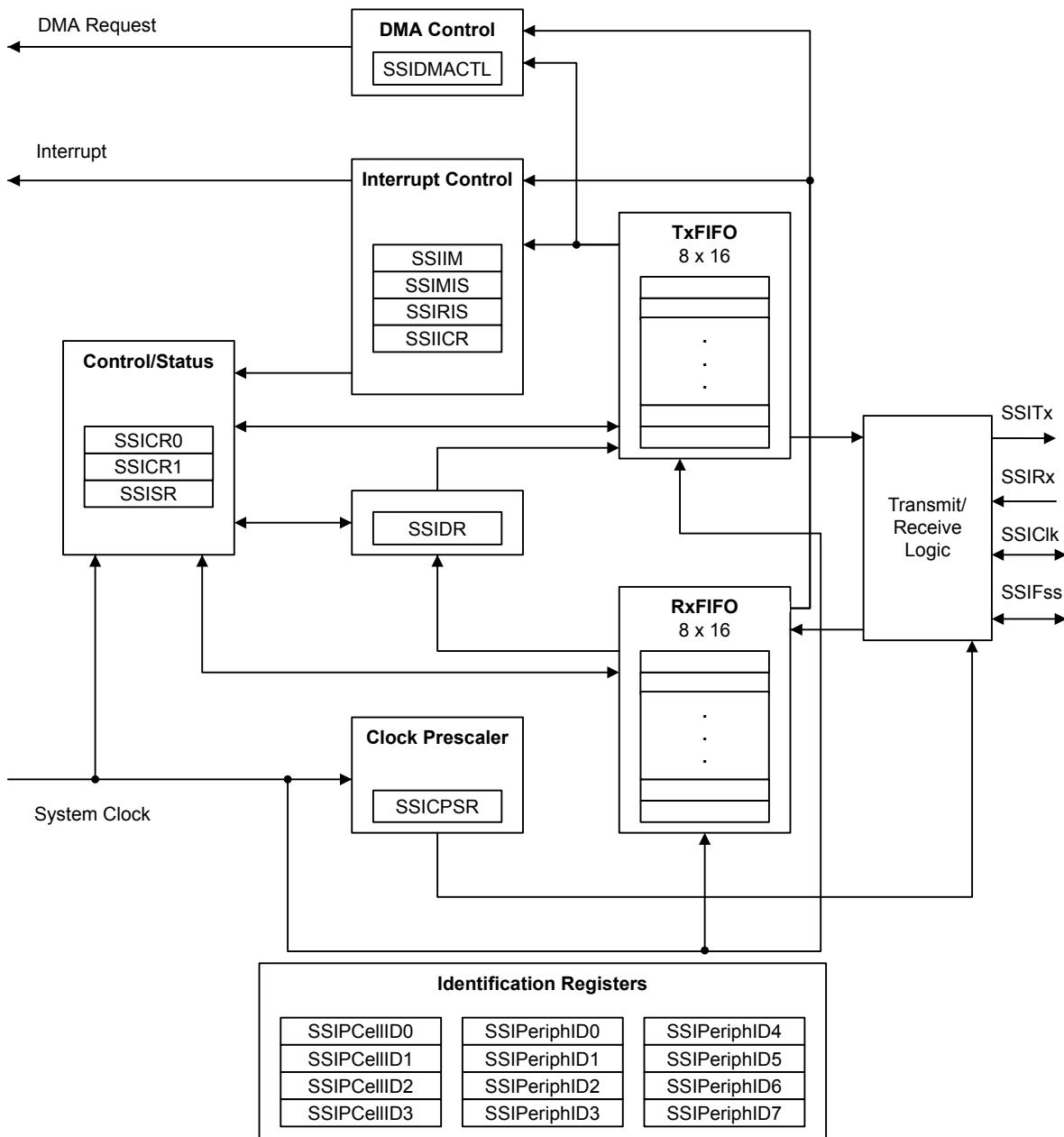
The Stellaris® microcontroller includes two Synchronous Serial Interface (SSI) modules. Each SSI is a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces.

The Stellaris LM3S9L97 controller includes two SSI modules with 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, each 16 bits wide and 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 (μ 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

14.1 Block Diagram

Figure 14-1. SSI Module Block Diagram



14.2 Signal Description

Table 14-1 on page 690 and Table 14-2 on page 690 list the external signals of the SSI module and describe the function of each. The SSI signals are alternate functions for some GPIO signals and default to be GPIO signals at reset., with the exception of the **SSI0Clk**, **SSI0Fss**, **SSI0Rx**, and **SSI0Tx** pins which default to the SSI function. The column in the table below titled "Pin Mux/Assignment" lists the possible GPIO pin placements for the SSI signals. The **AFSEL** bit in the **GPIOAFSEL** register (page 447) should be set to choose the SSI

function. The number in parentheses is the encoding that must be programmed into the `PMCN` field in the **GPIO Port Control (GPIOPCTL)** register (page 465) to assign the SSI signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 423.

Table 14-1. Signals for SSI (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
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	42 62 95	PF4 (9) PH6 (11) PE2 (2)	I	TTL	SSI module 1 receive.
SSI1Tx	15 41 96	PH7 (11) PF5 (9) PE3 (2)	O	TTL	SSI module 1 transmit.

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

Table 14-2. Signals for SSI (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
SSI0Clk	M4	PA2 (1)	I/O	TTL	SSI module 0 clock.
SSI0Fss	L4	PA3 (1)	I/O	TTL	SSI module 0 frame.
SSI0Rx	L5	PA4 (1)	I	TTL	SSI module 0 receive.
SSI0Tx	M5	PA5 (1)	O	TTL	SSI module 0 transmit.
SSI1Clk	J11 B11 B10	PF2 (9) PE0 (2) PH4 (11)	I/O	TTL	SSI module 1 clock.
SSI1Fss	J12 F10 A12	PF3 (9) PH5 (11) PE1 (2)	I/O	TTL	SSI module 1 frame.
SSI1Rx	K4 G3 A4	PF4 (9) PH6 (11) PE2 (2)	I	TTL	SSI module 1 receive.
SSI1Tx	H3 K3 B4	PH7 (11) PF5 (9) PE3 (2)	O	TTL	SSI module 1 transmit.

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

14.3 Functional Description

The SSI performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information. The transmit and receive paths are buffered with

internal FIFO memories allowing up to eight 16-bit values to be stored independently in both transmit and receive modes. The SSI also supports the µDMA interface. The transmit and receive FIFOs can be programmed as destination/source addresses in the µDMA module. µDMA operation is enabled by setting the appropriate bit(s) in the **SSIDMACTL** register (see page 718).

14.3.1 Bit Rate Generation

The SSI includes a programmable bit rate clock divider and prescaler to generate the serial output clock. Bit rates are supported to 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 (SysClk). 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 711). The clock is further divided by a value from 1 to 256, which is $1 + \text{SCR}$, where SCR is the value programmed in the **SSI Control 0 (SSICR0)** register (see page 704).

The frequency of the output clock SSIClk is defined by:

$$\text{SSIClk} = \text{SysClk} / (\text{CPSDVSR} * (1 + \text{SCR}))$$

Note: For master mode, the system clock must be at least two times faster than the SSIClk, with the restriction that SSIClk cannot be faster than 25 MHz. For slave mode, the system clock must be at least 12 times faster than the SSIClk.

See “Synchronous Serial Interface (SSI)” on page 1245 to view SSI timing parameters.

14.3.2 FIFO Operation

14.3.2.1 Transmit FIFO

The common transmit FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. The CPU writes data to the FIFO by writing the **SSI Data (SSIDR)** register (see page 708), and data is stored in the FIFO until it is read out by the transmission logic.

When configured as a master or a slave, parallel data is written into the transmit FIFO prior to serial conversion and transmission to the attached slave or master, respectively, through the SSITx pin.

In slave mode, the SSI transmits data each time the master initiates a transaction. If the transmit FIFO is empty and the master initiates, the slave transmits the 8th most recent value in the transmit FIFO. If less than 8 values have been written to the transmit FIFO since the SSI module clock was enabled using the **SSI** bit in the **RGCG1** register, then 0 is transmitted. Care should be taken to ensure that valid data is in the FIFO as needed. The SSI can be configured to generate an interrupt or a µDMA request when the FIFO is empty.

14.3.2.2 Receive FIFO

The common receive FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. Received data from the serial interface is stored in the buffer until read out by the CPU, which accesses the read FIFO by reading the **SSIDR** register.

When configured as a master or slave, serial data received through the SSIRx pin is registered prior to parallel loading into the attached slave or master receive FIFO, respectively.

14.3.3 Interrupts

The SSI can generate interrupts when the following conditions are observed:

- Transmit FIFO service (when the transmit FIFO is half full or less)

- Receive FIFO service (when the receive FIFO is half full or more)
- 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 generates a single interrupt request to the controller regardless of the number of active interrupts. Each of the four individual maskable interrupts can be masked by clearing the appropriate bit in the **SSI Interrupt Mask (SSIIM)** register (see page 712). Setting the appropriate mask bit enables the interrupt.

The individual outputs, along with a combined interrupt output, allow 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 713 and page 715, respectively).

The receive FIFO has a time-out period that is 32 periods at the rate of **SSIClk** (whether or not **SSIClk** is currently active) and is started when the RX FIFO goes from EMPTY to not-EMPTY. If the RX FIFO is emptied before 32 clocks have passed, the time-out period is reset. As a result, the ISR should clear the Receive FIFO Time-out Interrupt just after reading out the RX FIFO by writing a 1 to the **RTIC** bit in the **SSI Interrupt Clear (SSIICR)** register. The interrupt should not be cleared so late that the ISR returns before the interrupt is actually cleared, or the ISR may be re-activated unnecessarily.

The End-of-Transmission (EOT) interrupt indicates that the data has been transmitted completely. This interrupt can be used to indicate when it is safe to turn off the SSI module clock or enter sleep mode. In addition, because transmitted data and received data complete at exactly the same time, the interrupt can also indicate that read data is ready immediately, without waiting for the receive FIFO time-out period to complete.

14.3.4 Frame Formats

Each data frame is between 4 and 16 bits long, depending on the size of data programmed, and is transmitted starting with the MSB. There are three basic frame types that can be selected:

- Texas Instruments synchronous serial
- Freescale SPI
- MICROWIRE

For all three formats, the serial clock (**SSIClk**) is held inactive while the SSI is idle, and **SSIClk** transitions at the programmed frequency only during active transmission or reception of data. The idle state of **SSIClk** is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Freescale SPI and MICROWIRE frame formats, the serial frame (**SSIFss**) pin is active Low, and is asserted (pulled down) during the entire transmission of the frame.

For Texas Instruments synchronous serial frame format, the **SSIFss** pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format,

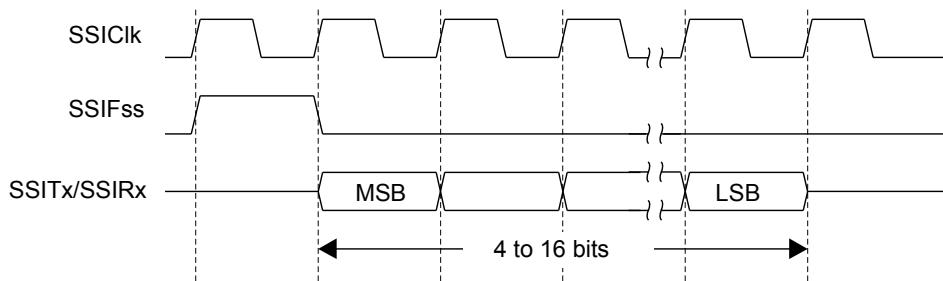
both the SSI and the off-chip slave device drive their output data on the rising edge of SSIClk and latch data from the other device on the falling edge.

Unlike the full-duplex transmission of the other two frame formats, the MICROWIRE format uses a special master-slave messaging technique which operates at half-duplex. In this mode, when a frame begins, an 8-bit control message is transmitted to the off-chip slave. During this transmit, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the requested data. The returned data can be 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

14.3.4.1 Texas Instruments Synchronous Serial Frame Format

Figure 14-2 on page 693 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

Figure 14-2. TI Synchronous Serial Frame Format (Single Transfer)

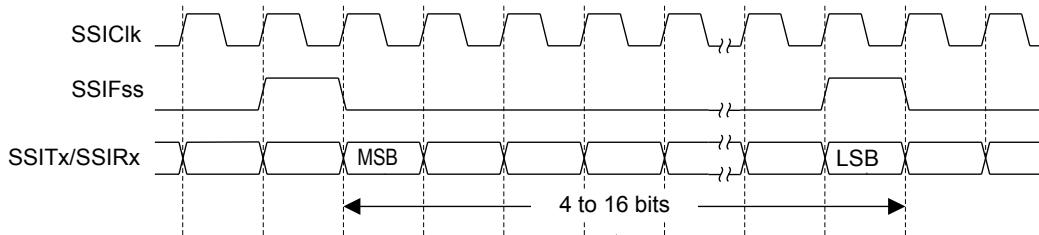


In this mode, SSIClk and SSIFss are forced Low, and the transmit data line SSITx is tristated whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, SSIFss is pulsed High for one SSIClk period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of SSIClk, the MSB of the 4 to 16-bit data frame is shifted out on the SSITx pin. Likewise, the MSB of the received data is shifted onto the SSIRx pin by the off-chip serial slave device.

Both the SSI and the off-chip serial slave device then clock each data bit into their serial shifter on each falling edge of SSIClk. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SSIClk after the LSB has been latched.

Figure 14-3 on page 694 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

Figure 14-3. TI Synchronous Serial Frame Format (Continuous Transfer)



14.3.4.2 Freescale SPI Frame Format

The Freescale SPI interface is a four-wire interface where the SSIFss signal behaves as a slave select. The main feature of the Freescale SPI format is that the inactive state and phase of the SSIClk signal are programmable through the SPO and SPH bits in the **SSISCR0** control register.

SPO Clock Polarity Bit

When the SPO clock polarity control bit is clear, it produces a steady state Low value on the SSIClk pin. If the SPO bit is set, a steady state High value is placed on the SSIClk pin when data is not being transferred.

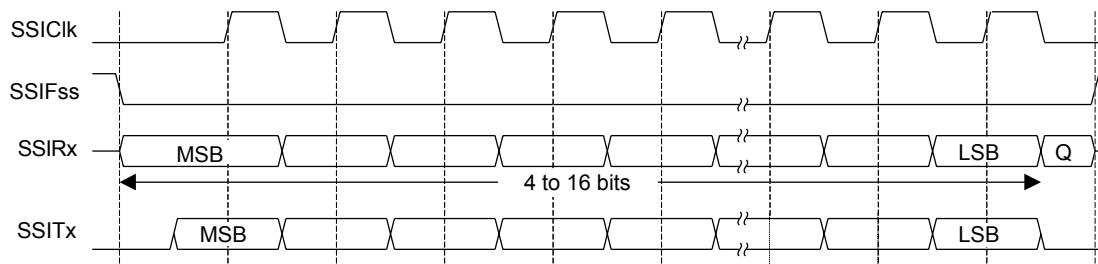
SPH Phase Control Bit

The SPH phase control bit selects the clock edge that captures data and allows it to change state. The state of this bit 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 clear, data is captured on the first clock edge transition. If the SPH bit is set, data is captured on the second clock edge transition.

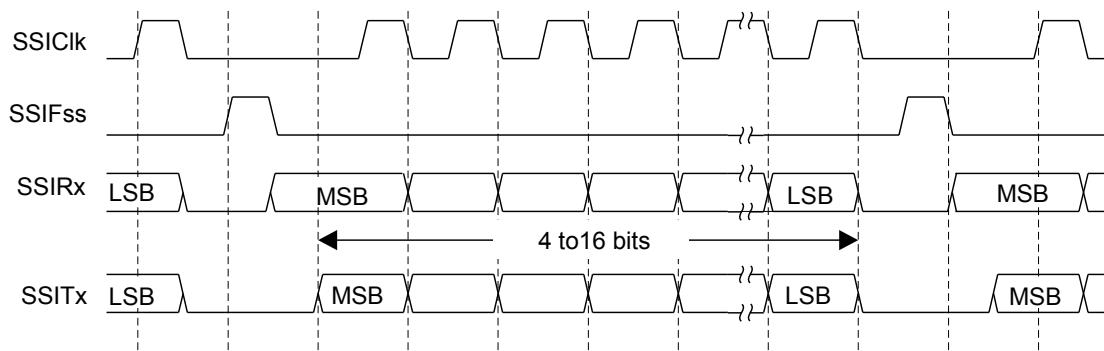
14.3.4.3 Freescale SPI Frame Format with SPO=0 and SPH=0

Single and continuous transmission signal sequences for Freescale SPI format with SPO=0 and SPH=0 are shown in Figure 14-4 on page 694 and Figure 14-5 on page 695.

Figure 14-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0



Note: Q is undefined.

Figure 14-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0

In this configuration, during idle periods:

- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low, causing 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. Once both the master and slave data have been set, the SSIClk master clock pin goes High after one additional half SSIClk period.

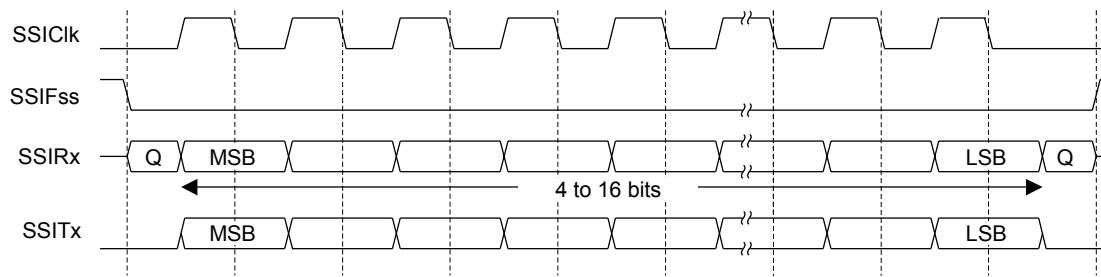
The data is now captured on the rising and propagated on the falling edges of the SSIClk signal.

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 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 clear. Therefore, the master device must raise the SSIFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSIFss pin is returned to its idle state one SSIClk period after the last bit has been captured.

14.3.4.4 Freescale SPI Frame Format with SPO=0 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=0 and SPH=1 is shown in Figure 14-6 on page 696, which covers both single and continuous transfers.

Figure 14-6. Freescale SPI Frame Format with SPO=0 and SPH=1

Note: Q is undefined.

In this configuration, during idle periods:

- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. The master SSITx output is enabled. After an additional one-half SSIClk period, both master and slave valid data are enabled onto their respective transmission lines. At the same time, the SSIClk is enabled with a rising edge transition.

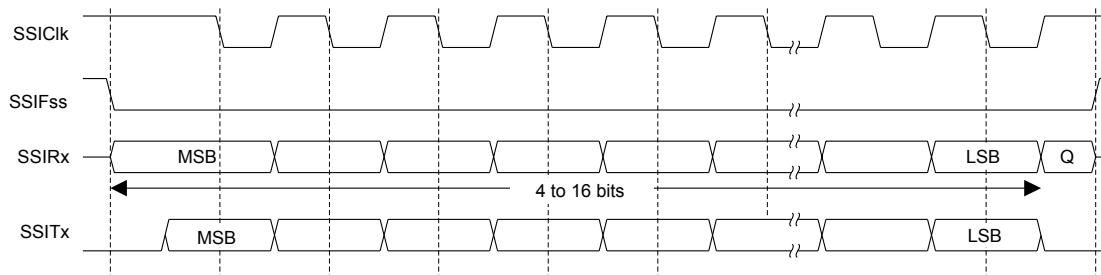
Data is then captured on the falling edges and propagated on the rising edges of the SSIClk signal.

In the case of a single word transfer, after all bits have been transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

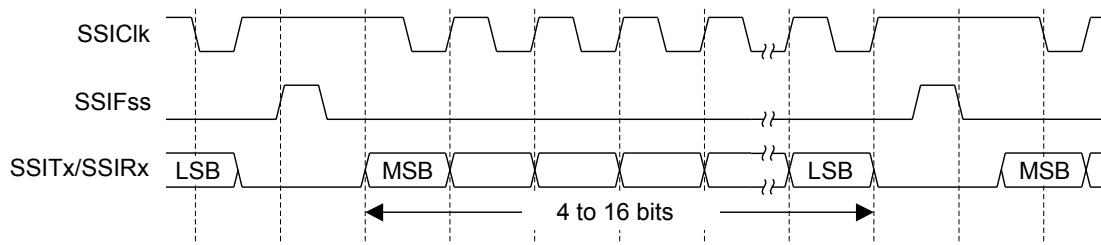
For continuous back-to-back transfers, the SSIFss pin is held Low between successive data words, and termination is the same as that of the single word transfer.

14.3.4.5 Freescale SPI Frame Format with SPO=1 and SPH=0

Single and continuous transmission signal sequences for Freescale SPI format with SPO=1 and SPH=0 are shown in Figure 14-7 on page 696 and Figure 14-8 on page 697.

Figure 14-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0

Note: Q is undefined.

Figure 14-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0

In this configuration, during idle periods:

- SSIClk is forced High
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low, causing slave data to be immediately transferred onto the SSITx line of the master. The master SSITx output pad is enabled.

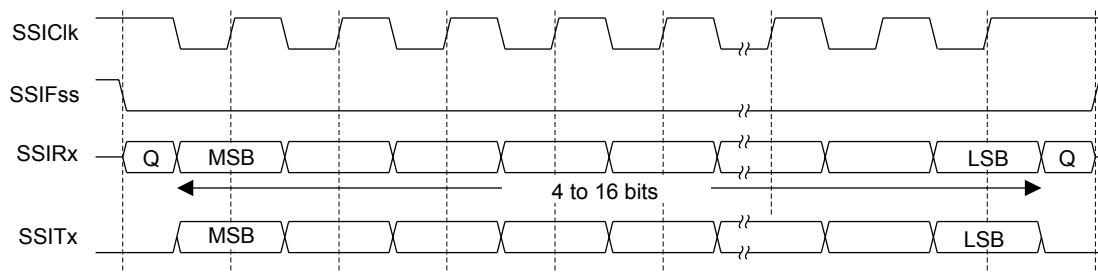
One-half period later, valid master data is transferred to the SSITx line. Once both the master and slave data have been set, the SSIClk master clock pin becomes Low after one additional half SSIClk period, meaning that data is captured on the falling edges and propagated on the rising edges of the SSIClk signal.

In the case of a single word transmission, after all bits of the data word are transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSIFss signal must be pulsed High between each data word transfer 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 clear. Therefore, the master device must raise the SSIFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSIFss pin is returned to its idle state one SSIClk period after the last bit has been captured.

14.3.4.6 Freescale SPI Frame Format with SPO=1 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=1 and SPH=1 is shown in Figure 14-9 on page 698, which covers both single and continuous transfers.

Figure 14-9. Freescale SPI Frame Format with SPO=1 and SPH=1

Note: Q is undefined.

In this configuration, during idle periods:

- SSIClk is forced High
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. The master SSITx output pad is enabled. After an additional one-half SSIClk period, both master and slave data are enabled onto their respective transmission lines. At the same time, SSIClk is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SSIClk signal.

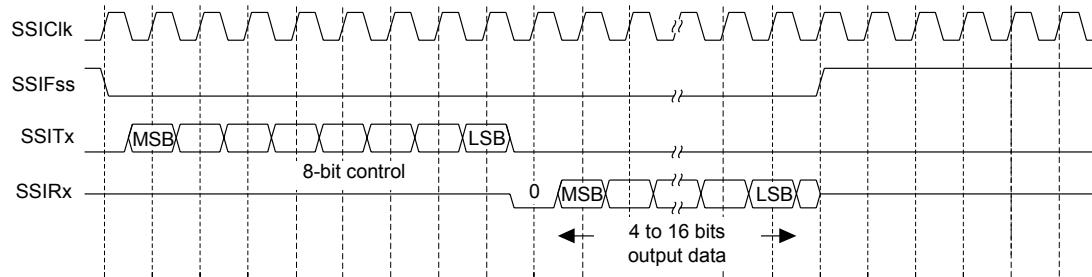
After all bits have been transferred, in the case of a single word transmission, the SSIFss line is returned to its idle high state one SSIClk period after the last bit has been captured.

For continuous back-to-back transmissions, the SSIFss pin remains in its active Low state until the final bit of the last word has been captured and then returns to its idle state as described above.

For continuous back-to-back transfers, the SSIFss pin is held Low between successive data words and termination is the same as that of the single word transfer.

14.3.4.7 MICROWIRE Frame Format

Figure 14-10 on page 698 shows the MICROWIRE frame format for a single frame. Figure 14-11 on page 699 shows the same format when back-to-back frames are transmitted.

Figure 14-10. MICROWIRE Frame Format (Single Frame)

MICROWIRE format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex and uses 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

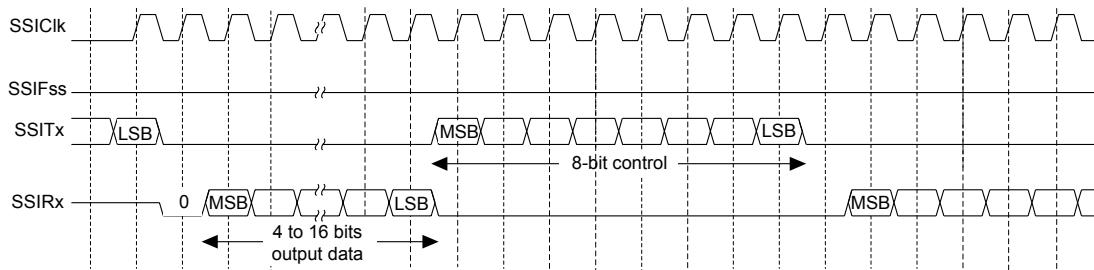
A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of SSIFss causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic and the MSB of the 8-bit control frame to be shifted out onto 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 each rising edge of 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, causing the data to be transferred to the receive FIFO.

Note: The off-chip slave device can tristate the receive line either on the falling edge of SSIClk after the LSB has been latched by the receive shifter or when the SSIFss pin goes High.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the SSIFss line is continuously asserted (held Low) and transmission of data occurs back-to-back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge of SSIClk, after the LSB of the frame has been latched into the SSI.

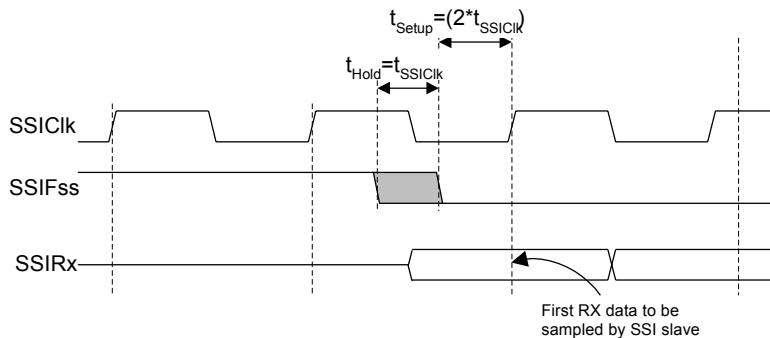
Figure 14-11. MICROWIRE Frame Format (Continuous Transfer)



In the MICROWIRE mode, the SSI slave samples the first bit of receive data on the rising edge of SSIClk after SSIFss has gone Low. Masters that drive a free-running SSIClk must ensure that the SSIFss signal has sufficient setup and hold margins with respect to the rising edge of SSIClk.

Figure 14-12 on page 700 illustrates these setup and hold time requirements. With respect to the SSIClk rising edge on which the first bit of receive data is to be sampled by the SSI slave, SSIFss must have a setup of at least two times the period of SSIClk on which the SSI operates. With respect to the SSIClk rising edge previous to this edge, SSIFss must have a hold of at least one SSIClk period.

Figure 14-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements



14.3.5 DMA Operation

The SSI peripheral provides an interface to the μDMA controller with separate channels for transmit and receive. The μDMA operation of the SSI is enabled through the **SSI DMA Control (SSIDMACTL)** register. When μDMA operation is enabled, the SSI asserts a μDMA 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 any data is 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 at least one empty location is in the transmit FIFO. The burst request is asserted whenever the transmit FIFO has 4 or more empty slots. The single and burst μDMA transfer requests are handled automatically by the μDMA controller depending how the μDMA channel is configured. To enable μDMA operation for the receive channel, the RXDMAE bit of the **DMA Control (SSIDMACTL)** register should be set. To enable μDMA operation for the transmit channel, the TXDMAE bit of **SSIDMACTL** should be set. If μDMA is enabled, then the μDMA controller triggers an interrupt when a transfer is complete. The interrupt occurs on the SSI interrupt vector. Therefore, if interrupts are used for SSI operation and μDMA is enabled, the SSI interrupt handler must be designed to handle the μDMA completion interrupt.

See “Micro Direct Memory Access (μDMA)” on page 365 for more details about programming the μDMA controller.

14.4 Initialization and Configuration

To enable and initialize the SSI, the following steps are necessary:

1. Enable the SSI module by setting the **SSI** bit in the **RCGC1** register (see page 276).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register (see page 285). To find out which GPIO port to enable, refer to Table 24-5 on page 1190.
3. Set the GPIO AFSEL bits for the appropriate pins (see page 447). To determine which GPIOs to configure, see Table 24-4 on page 1182.
4. Configure the **PMCn** fields in the **GPIOPCTL** register to assign the SSI signals to the appropriate pins. See page 465 and Table 24-5 on page 1190.

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 clear 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 365) 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:

$$\text{SSIClk} = \text{SysClk} / (\text{CPSDVSR} * (1 + \text{SCR})) \quad 1 \times 10^6 = 20 \times 10^6 / (\text{CPSDVSR} * (1 + \text{SCR}))$$

In this case, if **CPSDVSR**=0x2, **SCR** must be 0x9.

The configuration sequence would be as follows:

1. Ensure that the **SSE** bit in the **SSICR1** register is clear.
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.

14.5 Register Map

Table 14-3 on page 702 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 276).

Note: The SSI must be disabled (see the **SSE** bit in the **SSICR1** register) before any of the control registers are reprogrammed.

Table 14-3. SSI Register Map

Offset	Name	Type	Reset	Description	See page
0x000	SSICR0	R/W	0x0000.0000	SSI Control 0	704
0x004	SSICR1	R/W	0x0000.0000	SSI Control 1	706
0x008	SSIDR	R/W	0x0000.0000	SSI Data	708
0x00C	SSISR	RO	0x0000.0003	SSI Status	709
0x010	SSICPSR	R/W	0x0000.0000	SSI Clock Prescale	711
0x014	SSIIM	R/W	0x0000.0000	SSI Interrupt Mask	712
0x018	SSIRIS	RO	0x0000.0008	SSI Raw Interrupt Status	713
0x01C	SSIMIS	RO	0x0000.0000	SSI Masked Interrupt Status	715
0x020	SSIICR	W1C	0x0000.0000	SSI Interrupt Clear	717
0x024	SSIDMACTL	R/W	0x0000.0000	SSI DMA Control	718
0xFD0	SSIPeriphID4	RO	0x0000.0000	SSI Peripheral Identification 4	719
0xFD4	SSIPeriphID5	RO	0x0000.0000	SSI Peripheral Identification 5	720
0xFD8	SSIPeriphID6	RO	0x0000.0000	SSI Peripheral Identification 6	721
0xFDC	SSIPeriphID7	RO	0x0000.0000	SSI Peripheral Identification 7	722
0xFE0	SSIPeriphID0	RO	0x0000.0022	SSI Peripheral Identification 0	723
0xFE4	SSIPeriphID1	RO	0x0000.0000	SSI Peripheral Identification 1	724
0xFE8	SSIPeriphID2	RO	0x0000.0018	SSI Peripheral Identification 2	725
0xFEC	SSIPeriphID3	RO	0x0000.0001	SSI Peripheral Identification 3	726
0xFF0	SSIPCellID0	RO	0x0000.000D	SSI PrimeCell Identification 0	727
0xFF4	SSIPCellID1	RO	0x0000.00F0	SSI PrimeCell Identification 1	728
0xFF8	SSIPCellID2	RO	0x0000.0005	SSI PrimeCell Identification 2	729
0xFFC	SSIPCellID3	RO	0x0000.00B1	SSI PrimeCell Identification 3	730

14.6 Register Descriptions

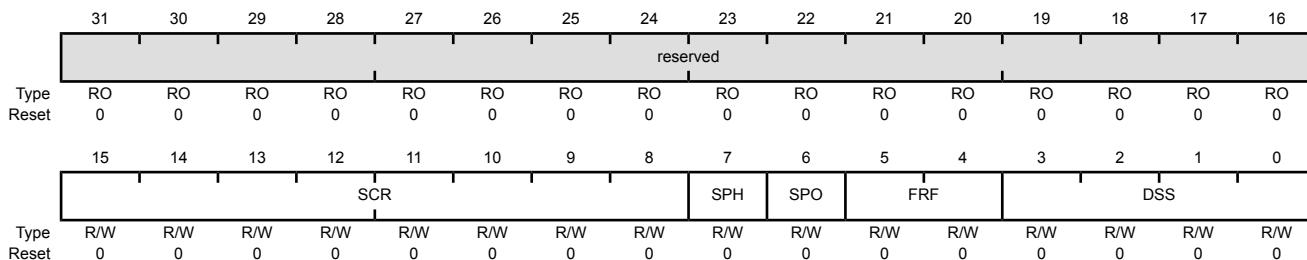
The remainder of this section lists and describes the SSI registers, in numerical order by address offset.

Register 1: SSI Control 0 (SSICR0), offset 0x000

The **SSICR0** register 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



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.
15:8	SCR	R/W	0x00	SSI Serial Clock Rate This bit field is used to generate the transmit and receive bit rate of the SSI. The bit rate is: $BR = SSIClk / (CPSDVSR * (1 + SCR))$ where CPSDVSR is an even value from 2-254 programmed in the SSICPSR register, and SCR is a value from 0-255.
7	SPH	R/W	0	SSI Serial Clock Phase This bit is only applicable to the Freescale SPI Format. The SPH control bit selects the clock edge that captures data and allows it to change state. This bit has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. Value Description 0 Data is captured on the first clock edge transition. 1 Data is captured on the second clock edge transition.
6	SPO	R/W	0	SSI Serial Clock Polarity Value Description 0 A steady state Low value is placed on the SSIClk pin. 1 A steady state High value is placed on the SSIClk pin when data is not being transferred.

Bit/Field	Name	Type	Reset	Description
5:4	FRF	R/W	0x0	SSI Frame Format Select
				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	0x0	SSI Data Size Select
				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

The **SSICR1** register 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															
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	0x0000.0	Software should not rely on the value of 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
		Value	Description	
		0	The TXRIS interrupt indicates that the transmit FIFO is half full or less.	
		1	The End of Transmit interrupt mode for the TXRIS interrupt is enabled.	
3	SOD	R/W	0	SSI Slave Mode Output Disable
		Value	Description	
		0	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 SSIT _x pin.	
		1	SSI can drive the SSIT _x output in Slave mode.	
		0	SSI must not drive the SSIT _x output in Slave mode.	
2	MS	R/W	0	SSI Master/Slave Select
		Value	Description	
		0	This bit selects Master or Slave mode and can be modified only when the SSI is disabled (SSE=0).	
		1	The SSI is configured as a master.	
		0	The SSI is configured as a slave.	

Bit/Field	Name	Type	Reset	Description
1	SSE	R/W	0	<p>SSI Synchronous Serial Port Enable</p> <p>Value Description</p> <p>0 SSI operation is disabled.</p> <p>1 SSI operation is enabled.</p> <p>Note: This bit must be cleared before any control registers are reprogrammed.</p>
0	LBM	R/W	0	<p>SSI Loopback Mode</p> <p>Value Description</p> <p>0 Normal serial port operation enabled.</p> <p>1 Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.</p>

Register 3: SSI Data (SSIDR), offset 0x008

Important: Use caution when reading this register. Performing a read may change bit status.

The **SSIDR** register is 16-bits wide. When the **SSIDR** register is read, the entry in the receive FIFO that is pointed to by the current FIFO read pointer is accessed. When a data value is removed by the SSI receive logic from the incoming data frame, it is placed into the entry in the receive FIFO pointed to by the current FIFO write pointer.

When the **SSIDR** register is written to, the entry in the transmit FIFO that is 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. Each data value 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 cleared, allowing the software to fill the transmit FIFO before enabling the SSI.

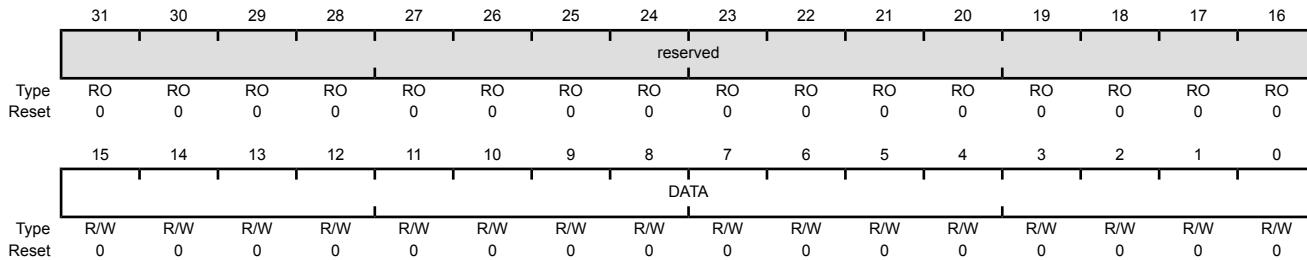
SSI Data (SSIDR)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0x008

Type R/W, reset 0x0000.0000



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

The **SSISR** register contains bits that indicate the FIFO fill status and the SSI busy status.

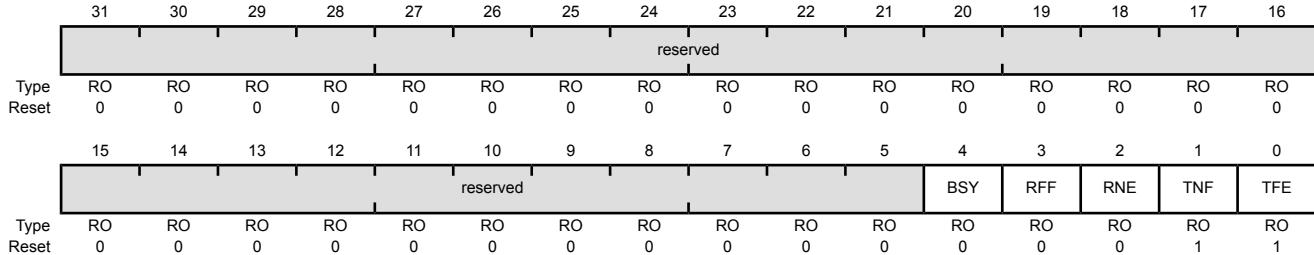
SSI Status (SSISR)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0x00C

Type RO, reset 0x0000.0003



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	BSY	RO	0	SSI Busy Bit
		Value	Description	
		0	The SSI is idle.	
		1	The SSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.	
3	RFF	RO	0	SSI Receive FIFO Full
		Value	Description	
		0	The receive FIFO is not full.	
		1	The receive FIFO is full.	
2	RNE	RO	0	SSI Receive FIFO Not Empty
		Value	Description	
		0	The receive FIFO is empty.	
		1	The receive FIFO is not empty.	
1	TNF	RO	1	SSI Transmit FIFO Not Full
		Value	Description	
		0	The transmit FIFO is full.	
		1	The transmit FIFO is not full.	

Bit/Field	Name	Type	Reset	Description
0	TFE	RO	1	SSI Transmit FIFO Empty
Value Description				
		0		The transmit FIFO is not empty.
		1		The transmit FIFO is empty.

Register 5: SSI Clock Prescale (SSICPSR), offset 0x010

The **SSICPSR** register specifies the division factor which is used to derive the SSIClk from the system clock. The clock is further divided by a value from 1 to 256, which is $1 + \text{SCR}$. SCR is programmed in the **SSICR0** register. The frequency of the SSIClk is defined by:

$$\text{SSIClk} = \text{SysClk} / (\text{CPSDVSR} * (1 + \text{SCR}))$$

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

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

On a read, this register gives the current value of the mask on the corresponding interrupt. Setting a bit sets the mask, preventing the interrupt from being signaled to the interrupt controller. Clearing a bit clears the corresponding mask, enabling the interrupt to be sent to the interrupt controller.

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															
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	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXIM	R/W	0	SSI Transmit FIFO Interrupt Mask Value Description 0 The transmit FIFO interrupt is masked. 1 The transmit FIFO interrupt is not masked.
2	RXIM	R/W	0	SSI Receive FIFO Interrupt Mask Value Description 0 The receive FIFO interrupt is masked. 1 The receive FIFO interrupt is not masked.
1	RTIM	R/W	0	SSI Receive Time-Out Interrupt Mask Value Description 0 The receive FIFO time-out interrupt is masked. 1 The receive FIFO time-out interrupt is not masked.
0	RORIM	R/W	0	SSI Receive Overrun Interrupt Mask Value Description 0 The receive FIFO overrun interrupt is masked. 1 The receive 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.

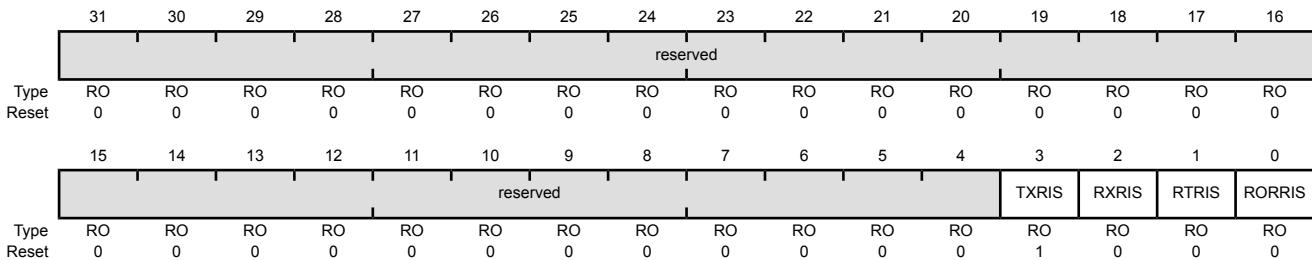
SSI Raw Interrupt Status (SSIRIS)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0x018

Type RO, reset 0x0000.0008



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	TXRIS	RO	1	SSI Transmit FIFO Raw Interrupt Status Value Description 0 No interrupt. 1 If the EOT bit in the SSICR1 register is clear, the transmit FIFO is half full or less. If the EOT bit is set, the transmit FIFO is empty, and the last bit has been transmitted out of the serializer. This bit is cleared when the transmit FIFO is more than half full (if the EOT bit is clear) or when it has any data in it (if the EOT bit is set).
2	RXRIS	RO	0	SSI Receive FIFO Raw Interrupt Status Value Description 0 No interrupt. 1 The receive FIFO is half full or more. This bit is cleared when the receive FIFO is less than half full.
1	RTRIS	RO	0	SSI Receive Time-Out Raw Interrupt Status Value Description 0 No interrupt. 1 The receive time-out has occurred. This bit is cleared when a 1 is written to the RTIC bit in the SSI Interrupt Clear (SSICR) register.

Bit/Field	Name	Type	Reset	Description
0	RORRIS	RO	0	SSI Receive Overrun Raw Interrupt Status
Value Description				
0 No interrupt.				
1 The receive FIFO has overflowed				
This bit is cleared when a 1 is written to the RORIC bit in the SSI Interrupt Clear (SSIICR) register.				

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															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXMIS	RO	0	SSI Transmit FIFO Masked Interrupt Status
	Value Description			
	0	An interrupt has not occurred or is masked.		
	1	An unmasked interrupt was signaled due to the transmit FIFO being half full or less (if the EOT bit is clear) or due to the transmission of the last data bit (if the EOT bit is set).		
	This bit is cleared when the transmit FIFO is more than half full (if the EOT bit is clear) or when it has any data in it (if the EOT bit is set).			
2	RXMISS	RO	0	SSI Receive FIFO Masked Interrupt Status
	Value Description			
	0	An interrupt has not occurred or is masked.		
	1	An unmasked interrupt was signaled due to the receive FIFO being half full or less.		
	This bit is cleared when the receive FIFO is less than half full.			
1	RTMIS	RO	0	SSI Receive Time-Out Masked Interrupt Status
	Value Description			
	0	An interrupt has not occurred or is masked.		
	1	An unmasked interrupt was signaled due to the receive time out.		
	This bit is cleared when a 1 is written to the RTIC bit in the SSI Interrupt Clear (SSIICR) register.			

Bit/Field	Name	Type	Reset	Description
0	RORMIS	RO	0	SSI Receive Overrun Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to the receive FIFO overflowing.
This bit is cleared when a 1 is written to the RORIC bit in the SSI Interrupt Clear (SSIIICR) register.				

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															
Type	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

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	RTIC	W1C	0	SSI Receive Time-Out Interrupt Clear Writing a 1 to this bit clears the RTRIS bit in the SSIRIS register and the RTMIS bit in the SSIMIS register.
0	RORIC	W1C	0	SSI Receive Overrun Interrupt Clear Writing a 1 to this bit clears the RRRIS bit in the SSIRIS register and the RORMIS bit in the SSIMIS register.

Register 10: SSI DMA Control (SSIDMACTL), offset 0x024

The **SSIDMACTL** register is the µDMA control register.

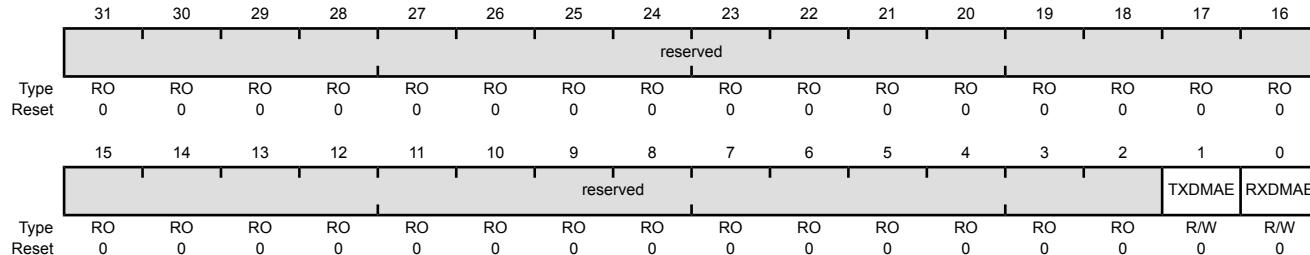
SSI DMA Control (SSIDMACTL)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0x024

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	TXDMAE	R/W	0	Transmit DMA Enable Value Description 0 µDMA for the transmit FIFO is disabled. 1 µDMA for the transmit FIFO is enabled.
0	RXDMAE	R/W	0	Receive DMA Enable Value Description 0 µDMA for the receive FIFO is disabled. 1 µDMA 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.

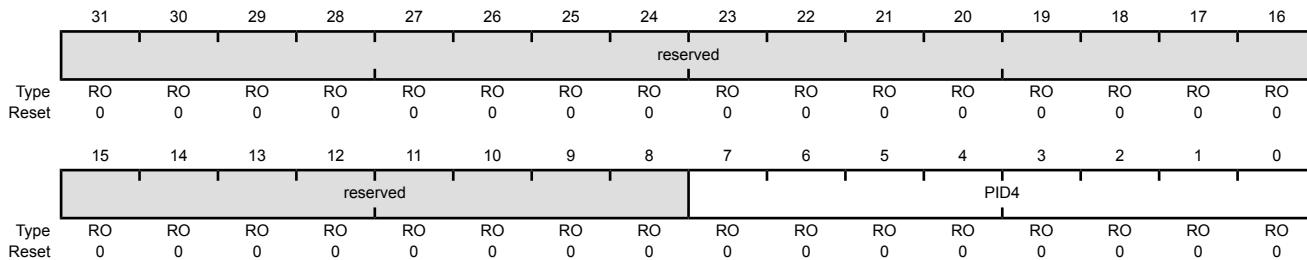
SSI Peripheral Identification 4 (SSIPeriphID4)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0xFD0

Type RO, reset 0x0000.0000



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	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.

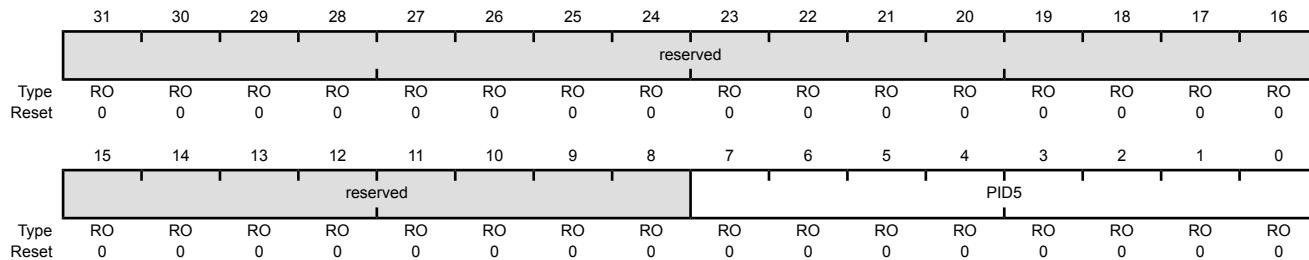
SSI Peripheral Identification 5 (SSIPeriphID5)

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	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	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.

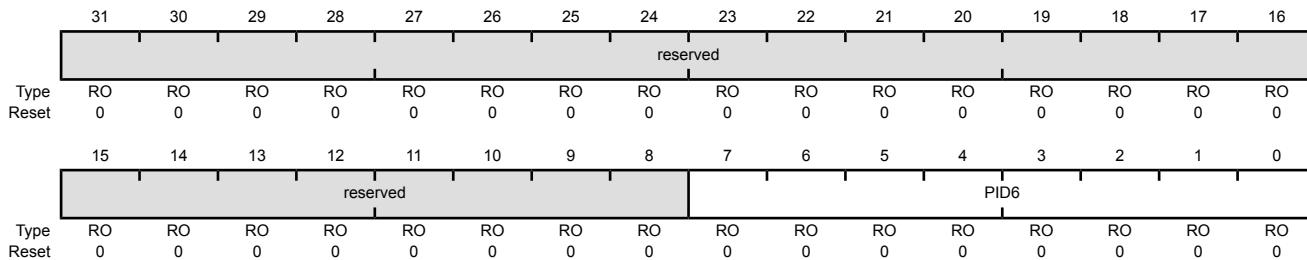
SSI Peripheral Identification 6 (SSIPeriphID6)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0xFD8

Type RO, reset 0x0000.0000



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	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.

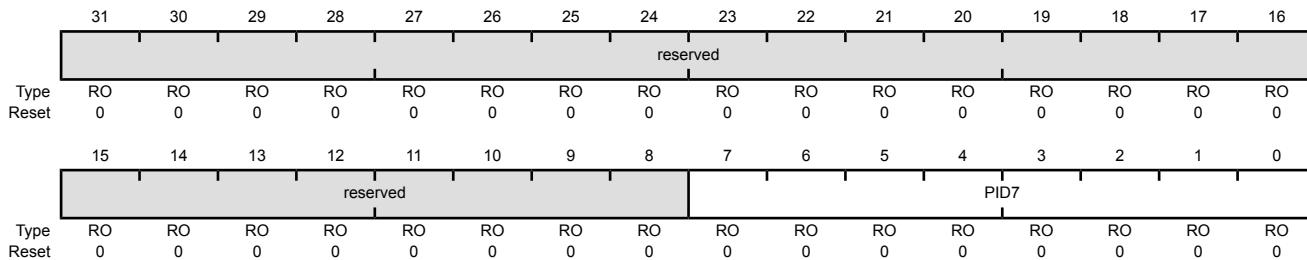
SSI Peripheral Identification 7 (SSIPeriphID7)

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	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	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.

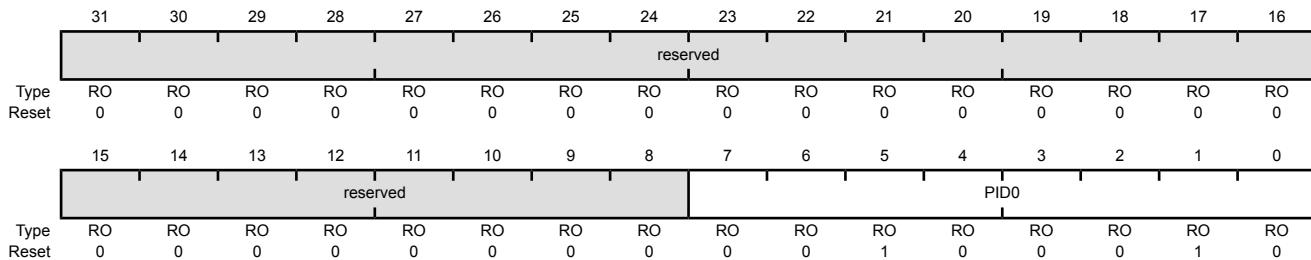
SSI Peripheral Identification 0 (SSIPeriphID0)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0xFE0

Type RO, reset 0x0000.0022



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	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.

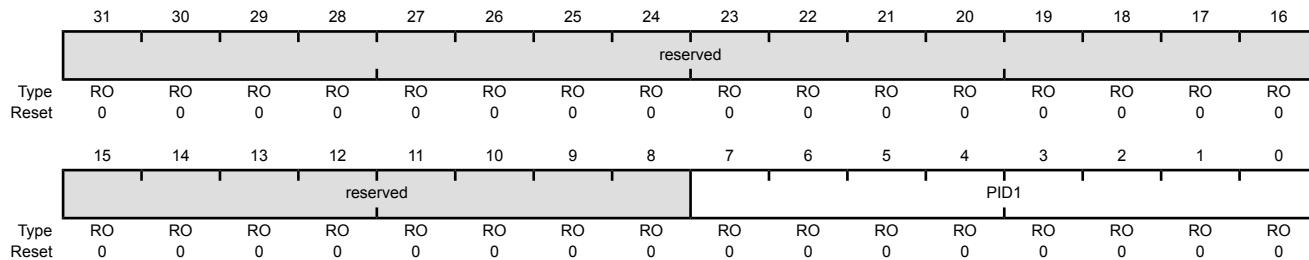
SSI Peripheral Identification 1 (SSIPeriphID1)

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	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	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.

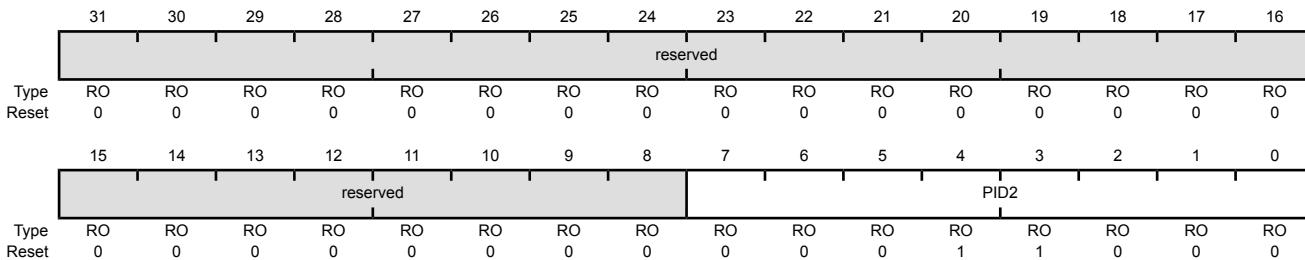
SSI Peripheral Identification 2 (SSIPeriphID2)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0xFE8

Type RO, reset 0x0000.0018



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	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.

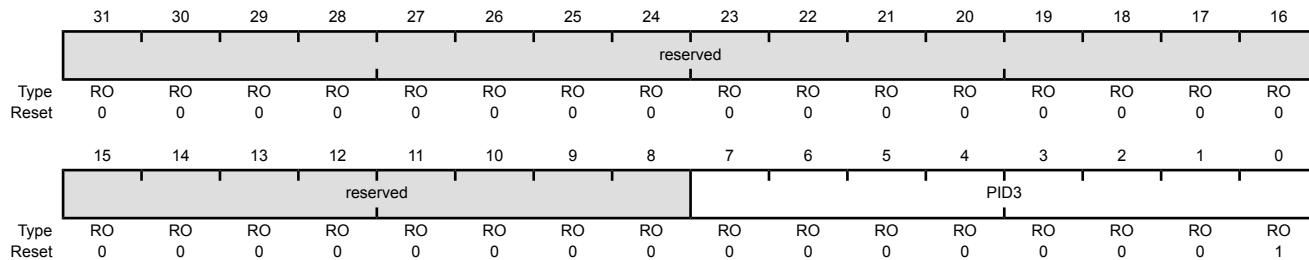
SSI Peripheral Identification 3 (SSIPeriphID3)

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	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	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.

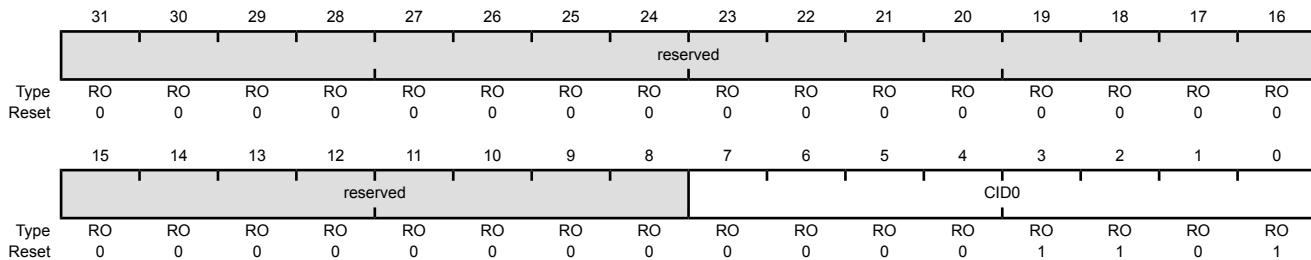
SSI PrimeCell Identification 0 (SSIPCellID0)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0xFF0

Type RO, reset 0x0000.000D



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	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.

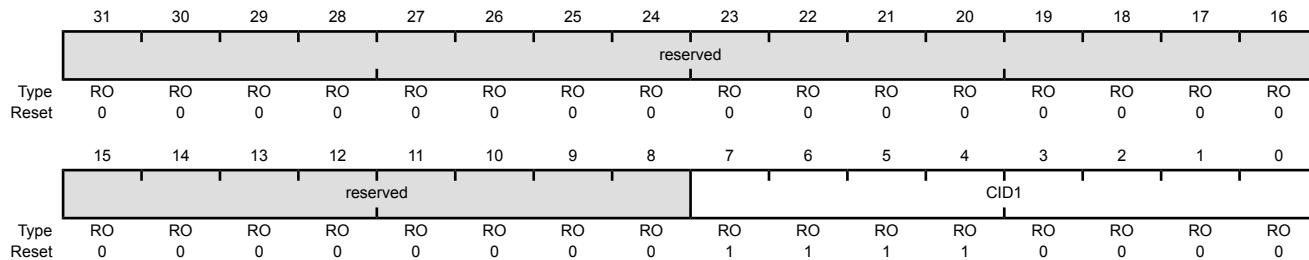
SSI PrimeCell Identification 1 (SSIPCellID1)

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	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	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.

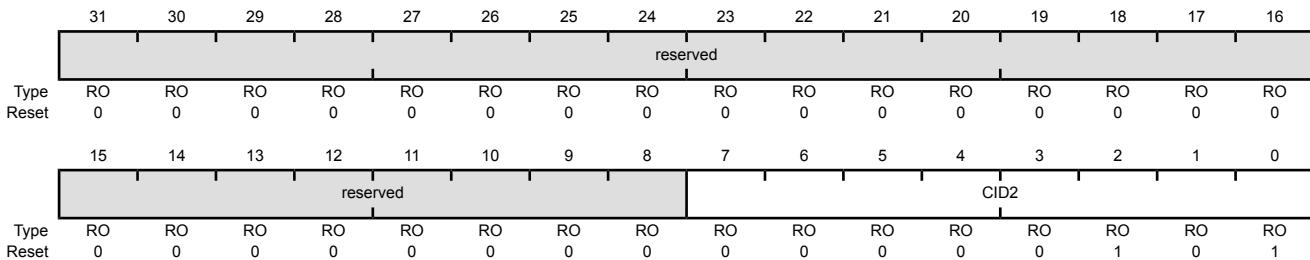
SSI PrimeCell Identification 2 (SSIPCellID2)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0xFF8

Type RO, reset 0x0000.0005



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	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.

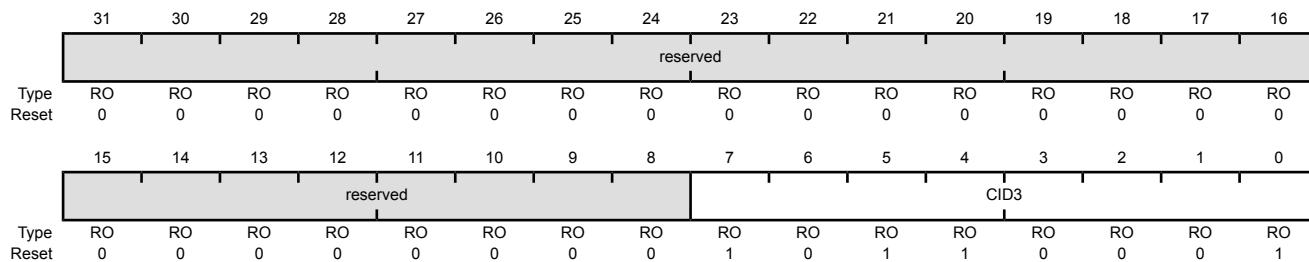
SSI PrimeCell Identification 3 (SSIPCellID3)

SSI0 base: 0x4000.8000

SSI1 base: 0x4000.9000

Offset 0xFFC

Type RO, reset 0x0000.00B1



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	SSI PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

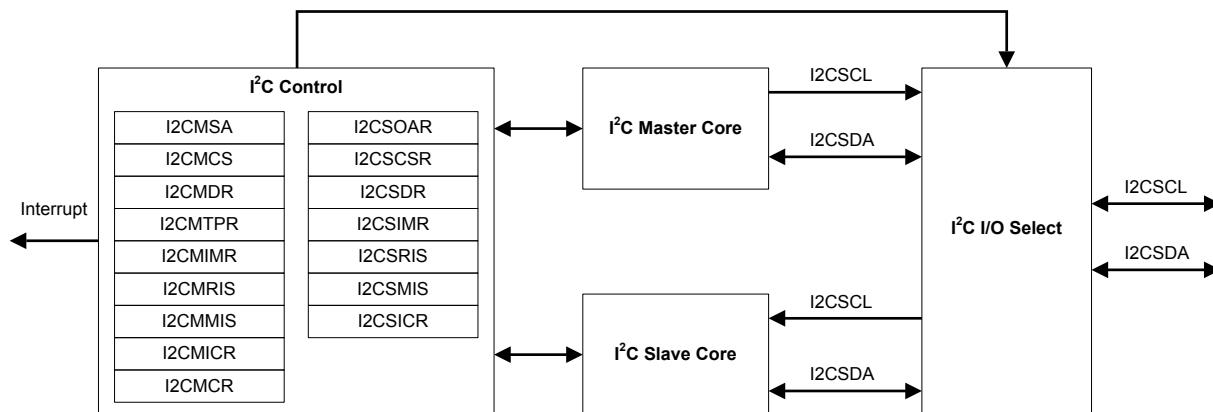
15 Inter-Integrated Circuit (I²C) Interface

The Inter-Integrated Circuit (I²C) 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 I²C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I²C bus may also be used for system testing and diagnostic purposes in product development and manufacture. The LM3S9L97 microcontroller includes two I²C modules, providing the ability to interact (both transmit and receive) with other I²C devices on the bus.

The Stellaris® LM3S9L97 controller includes two I²C modules with the following features:

- Devices on the I²C bus can be designated as either a master or a slave
 - Supports both transmitting and receiving data as either a master or a slave
 - Supports simultaneous master and slave operation
- Four I²C 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 transferred 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

15.1 Block Diagram

Figure 15-1. I^2C Block Diagram

15.2 Signal Description

Table 15-1 on page 732 and Table 15-2 on page 732 list the external signals of the I^2C interface and describe the function of each. The I^2C interface signals are alternate functions for some GPIO signals and default to be GPIO signals at reset., with the exception of the I^2C0SCL and I^2C0SDA pins which default to the I^2C function. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the I^2C signals. The **AFSEL** bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 447) should be set to choose the I^2C function. The number in parentheses is the encoding that must be programmed into the $PMCn$ field in the **GPIO Port Control (GPIOPCTL)** register (page 465) to assign the I^2C signal to the specified GPIO port pin. Note that the I^2C pins should be set to open drain using the **GPIO Open Drain Select (GPIOODR)** register. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 423.

Table 15-1. Signals for I^2C (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
I^2C0SCL	72	PB2 (1)	I/O	OD	I^2C module 0 clock.
I^2C0SDA	65	PB3 (1)	I/O	OD	I^2C module 0 data.
I^2C1SCL	14 19 26 34	PJ0 (11) PG0 (3) PA0 (8) PA6 (1)	I/O	OD	I^2C module 1 clock.
I^2C1SDA	18 27 35 87	PG1 (3) PA1 (8) PA7 (1) PJ1 (11)	I/O	OD	I^2C module 1 data.

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

Table 15-2. Signals for I^2C (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
I^2C0SCL	A11	PB2 (1)	I/O	OD	I^2C module 0 clock.

Table 15-2. Signals for I²C (108BGA) (continued)

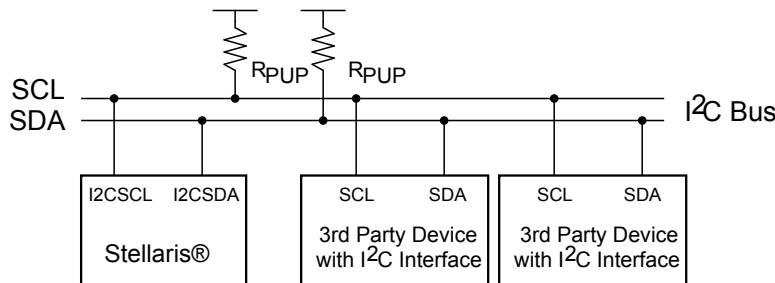
Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
I ₂ C0SDA	E11	PB3 (1)	I/O	OD	I ² C module 0 data.
I ₂ C1SCL	F3 K1 L3 L6	PJ0 (11) PG0 (3) PA0 (8) PA6 (1)	I/O	OD	I ² C module 1 clock.
I ₂ C1SDA	K2 M3 M6 B6	PG1 (3) PA1 (8) PA7 (1) PJ1 (11)	I/O	OD	I ² C module 1 data.

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

15.3 Functional Description

Each I²C 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 I²C bus configuration is shown in Figure 15-2.

See “Inter-Integrated Circuit (I²C) Interface” on page 1246 for I²C timing diagrams.

Figure 15-2. I²C Bus Configuration

15.3.1 I²C Bus Functional Overview

The I²C bus uses only two signals: SDA and SCL, named I₂CSCL and I₂CSDA on Stellaris microcontrollers. SDA is the bi-directional serial data line and SCL is the bi-directional serial clock line. The bus is considered idle when both lines are High.

Every transaction on the I²C 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 733) is unrestricted, but each byte has to be followed by an acknowledge bit, and data must be transferred MSB first. When a receiver cannot receive another complete byte, it can hold the clock line SCL Low and force the transmitter into a wait state. The data transfer continues when the receiver releases the clock SCL.

15.3.1.1 START and STOP Conditions

The protocol of the I²C bus defines two states to begin and end a transaction: START and STOP. A High-to-Low transition on the SDA line while the SCL is High is defined as a START condition, and a Low-to-High transition on the SDA line while SCL is High is defined as a STOP condition. The bus is considered busy after a START condition and free after a STOP condition. See Figure 15-3.

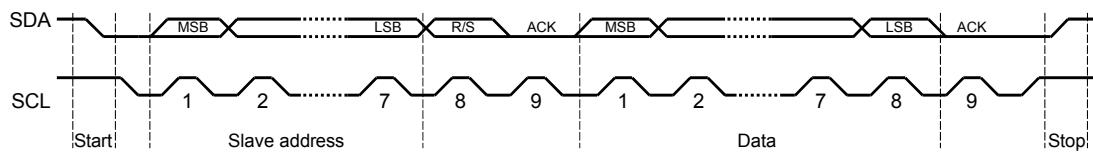
Figure 15-3. START and STOP Conditions

The STOP bit determines if the cycle stops at the end of the data cycle or continues on to a repeated START condition. To generate a single transmit cycle, the **I²C Master Slave Address (I2CMSA)** register is written with the desired address, the R/S bit is cleared, and the Control register is written with ACK=X (0 or 1), STOP=1, START=1, and RUN=1 to perform the operation and stop. When the operation is completed (or aborted due to an error), the interrupt pin becomes active and the data may be read from the **I2CMDR** register. When the I²C module operates in Master receiver mode, the ACK bit is normally set causing the I²C bus controller to transmit an acknowledge automatically after each byte. This bit must be cleared when the I²C bus controller requires no further data to be transmitted from the slave transmitter.

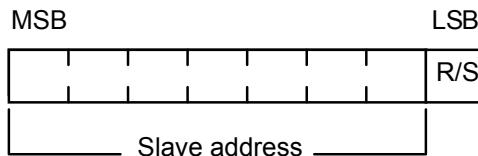
When operating in slave mode, two bits in the **I2CSRIS** register indicate detection of start and stop conditions on the bus; while two bits in the **I2CSMIS** register allow start and stop conditions to be promoted to controller interrupts (when interrupts are enabled).

15.3.1.2 Data Format with 7-Bit Address

Data transfers follow the format shown in Figure 15-4. After the START condition, a slave address is transmitted. This address is 7-bits long followed by an eighth bit, which is a data direction bit (R/S bit in the **I2CMSA** register). If the R/S bit is clear, it indicates a transmit operation (send), and if it is set, it 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/transmit formats are then possible within a single transfer.

Figure 15-4. Complete Data Transfer with a 7-Bit Address

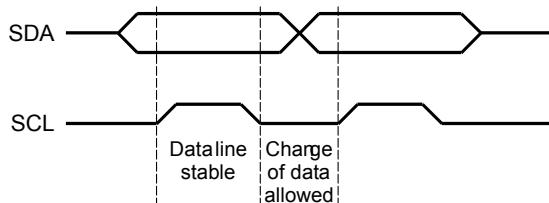
The first seven bits of the first byte make up the slave address (see Figure 15-5). The eighth bit determines the direction of the message. A zero in the R/S position of the first byte means that the master transmits (sends) data to the selected slave, and a one in this position means that the master receives data from the slave.

Figure 15-5. R/S Bit in First Byte

15.3.1.3 Data Validity

The data on the SDA line must be stable during the high period of the clock, and the data line can only change when SCL is Low (see Figure 15-6).

Figure 15-6. Data Validity During Bit Transfer on the I²C Bus



15.3.1.4 Acknowledge

All bus transactions have a required acknowledge clock cycle that is generated by the master. During the acknowledge cycle, the transmitter (which can be the master or slave) releases the SDA line. To acknowledge the transaction, the receiver must pull down SDA during the acknowledge clock cycle. The data transmitted out by the receiver during the acknowledge cycle must comply with the data validity requirements described in “Data Validity” on page 735.

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. Because the master controls the number of bytes in the transfer, it signals the end of data to the slave transmitter by not generating an acknowledge on the last data byte. The slave transmitter must then release SDA to allow the master to generate the STOP or a repeated START condition.

15.3.1.5 Arbitration

A master may start a transfer only if the bus is idle. It's possible for two or more masters to generate a START condition within minimum hold time of the START condition. In these situations, an arbitration scheme takes place on the SDA line, while SCL is High. During arbitration, the first of the competing master devices to place a '1' (High) on SDA while another master transmits a '0' (Low) switches off its data output stage and retires until the bus is idle again.

Arbitration can take place over several bits. Its first stage is a comparison of address bits, and if both masters are trying to address the same device, arbitration continues on to the comparison of data bits.

15.3.2 Available Speed Modes

The I²C bus can run in either Standard mode (100 kbps) or Fast mode (400 kbps). The selected mode should match the speed of the other I²C devices on the bus.

15.3.2.1 Standard and Fast Modes

Standard and Fast modes are selected using a value in the **I²C Master Timer Period (I2CMTPR)** register that results in an SCL frequency of 100 kbps for Standard mode or 400 kbps for Fast mode.

The I²C 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 **I2CMTPR** register (see page 753).

The I²C clock period is calculated as follows:

$$SCL_PERIOD = 2 \times (1 + TIMER_PRD) \times (SCL_LP + SCL_HP) \times CLK_PRD$$

For example:

$$CLK_PRD = 50 \text{ ns}$$

$$TIMER_PRD = 2$$

$$SCL_LP = 6$$

$$SCL_HP = 4$$

yields a SCL frequency of:

$$1/SCL_PERIOD = 333 \text{ KHz}$$

Table 15-3 gives examples of the timer periods that should be used to generate both Standard and Fast mode SCL frequencies based on various system clock frequencies.

Table 15-3. Examples of I²C 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
80 MHz	0x27	100 Kbps	0x09	400 Kbps

15.3.3 Interrupts

The I²C can generate interrupts when the following conditions are observed:

- Master transaction completed
- Master arbitration lost
- Master transaction error
- Slave transaction received
- Slave transaction requested
- Stop condition on bus detected
- Start condition on bus detected

The I²C master and I²C slave modules have separate interrupt signals. While both modules can generate interrupts for multiple conditions, only a single interrupt signal is sent to the interrupt controller.

15.3.3.1 I²C Master Interrupts

The I²C master module generates an interrupt when a transaction completes (either transmit or receive), when arbitration is lost, or when an error occurs during a transaction. To enable the I²C master interrupt, software must set the `IM` bit in the **I²C Master Interrupt Mask (I2CMIMR)** register. When an interrupt condition is met, software must check the `ERROR` and `ARBLST` bits in the **I²C Master Control/Status (I2CMCS)** register to verify that an error didn't occur during the last transaction and to ensure that arbitration has not been lost. An error condition is asserted if the last transaction wasn't acknowledged by the slave. If an error is not detected and the master has not lost arbitration, the application can proceed with the transfer. The interrupt is cleared by writing a 1 to the `IC` bit in the **I²C Master Interrupt Clear (I2CMICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I²C Master Raw Interrupt Status (I2CMRIS)** register.

15.3.3.2 I²C Slave Interrupts

The slave module can generate an interrupt when data has been received or requested. This interrupt is enabled by setting the `DATAIM` bit in the **I²C Slave Interrupt Mask (I2CSIMR)** register. Software determines whether the module should write (transmit) or read (receive) data from the **I²C Slave Data (I2CSDR)** register, by checking the `RREQ` and `TREQ` bits of the **I²C 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 setting the `DATAIC` bit in the **I²C 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 setting the `STARTIM` and `STOPIM` bits of the **I²C Slave Interrupt Mask (I2CSIMR)** register and cleared by writing a 1 to the `STOPIC` and `STARTIC` bits of the **I²C Slave Interrupt Clear (I2CSICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I²C Slave Raw Interrupt Status (I2CSRIS)** register.

15.3.4 Loopback Operation

The I²C modules can be placed into an internal loopback mode for diagnostic or debug work by setting the `LPBK` bit in the **I²C Master Configuration (I2CMCR)** register. In loopback mode, the SDA and SCL signals from the master and slave modules are tied together.

15.3.5 Command Sequence Flow Charts

This section details the steps required to perform the various I²C transfer types in both master and slave mode.

15.3.5.1 I²C Master Command Sequences

The figures that follow show the command sequences available for the I²C master.

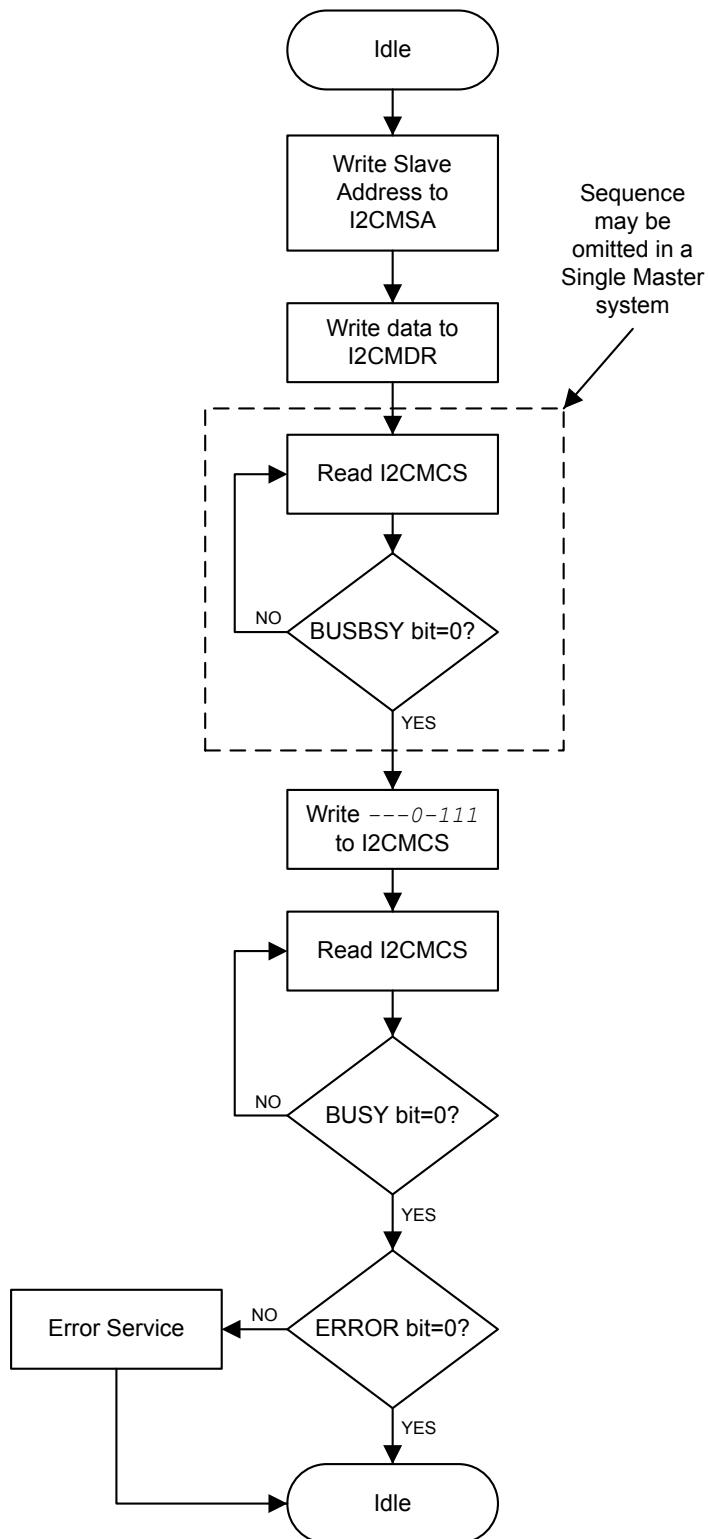
Figure 15-7. Master Single TRANSMIT

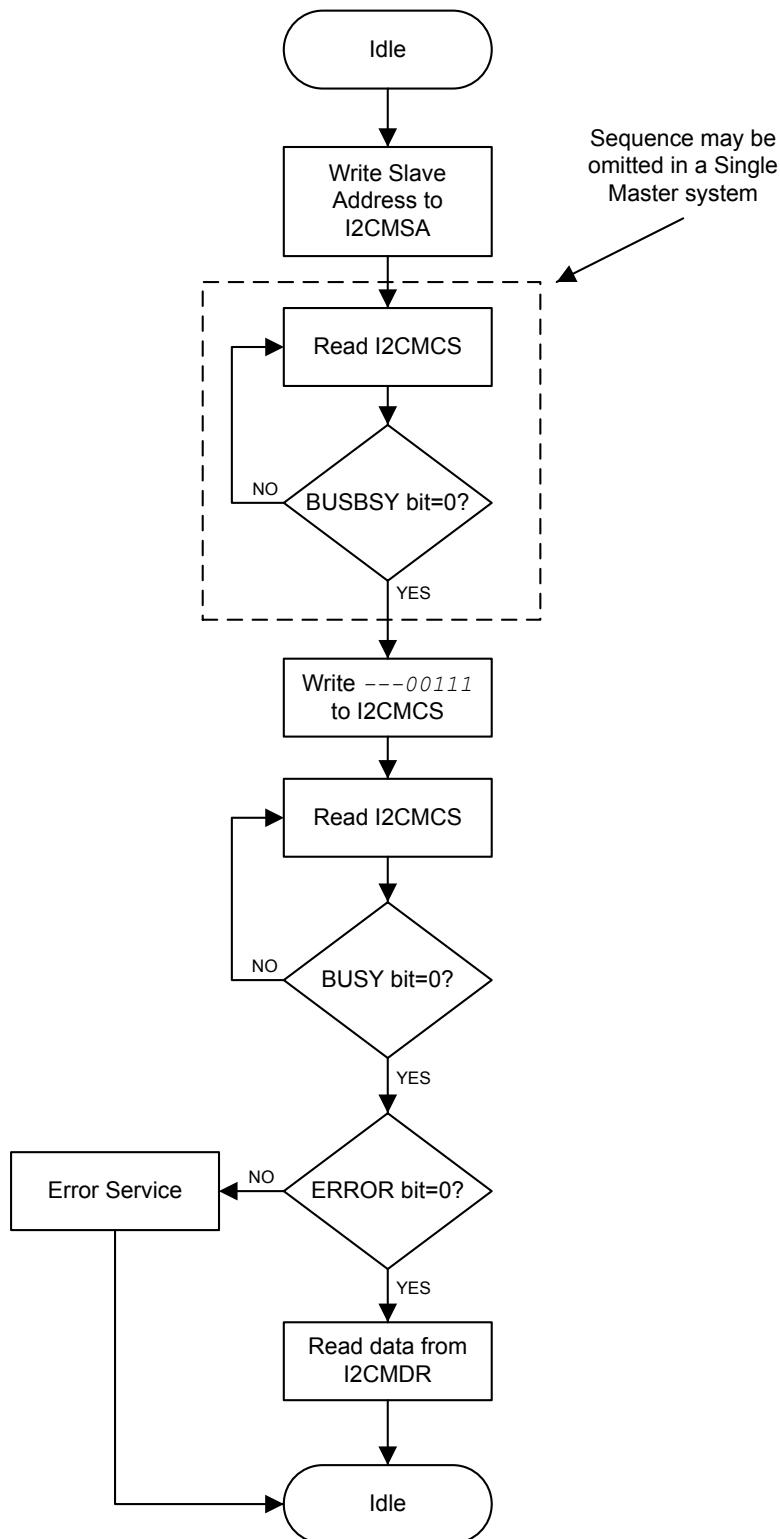
Figure 15-8. Master Single RECEIVE

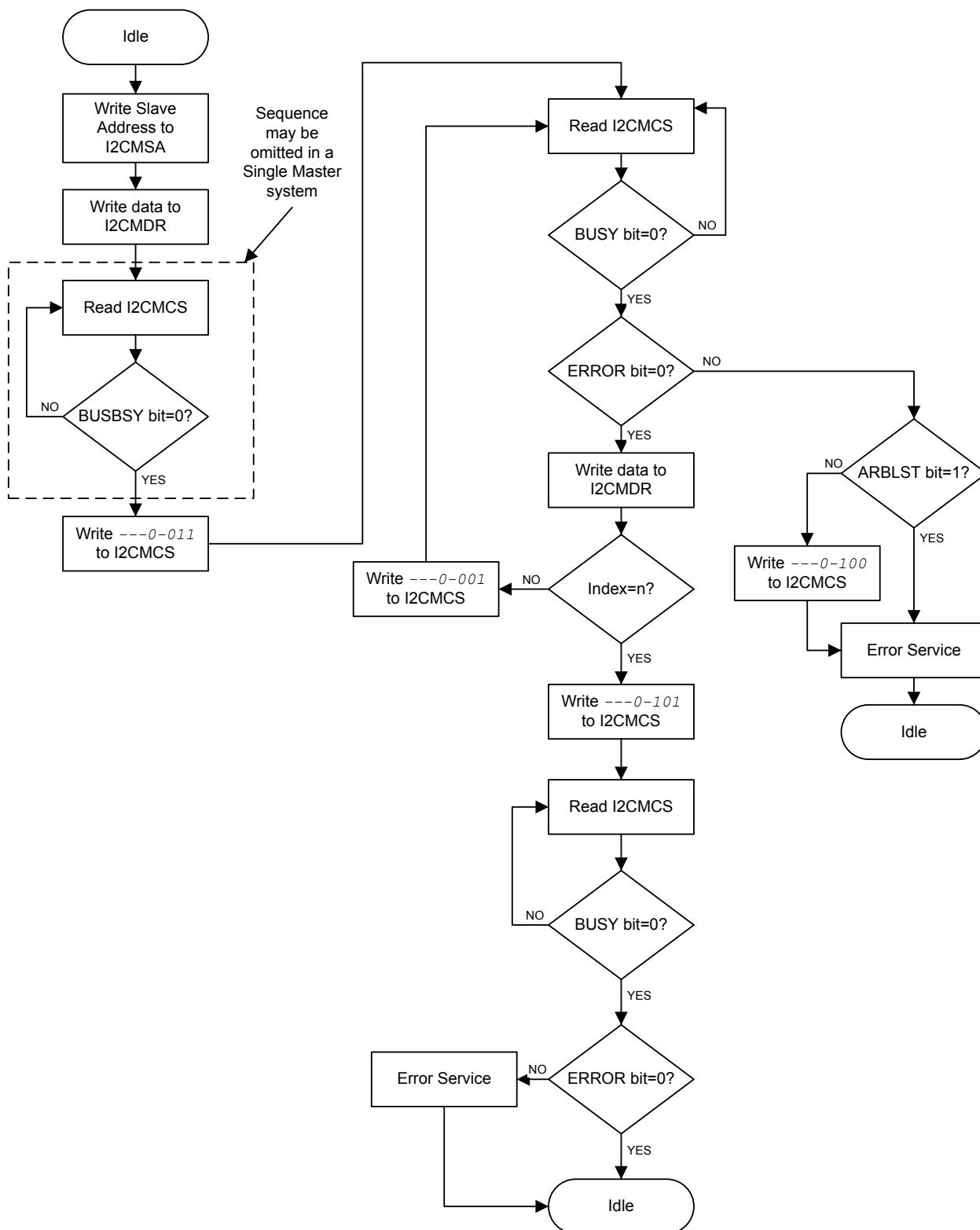
Figure 15-9. Master TRANSMIT with Repeated START

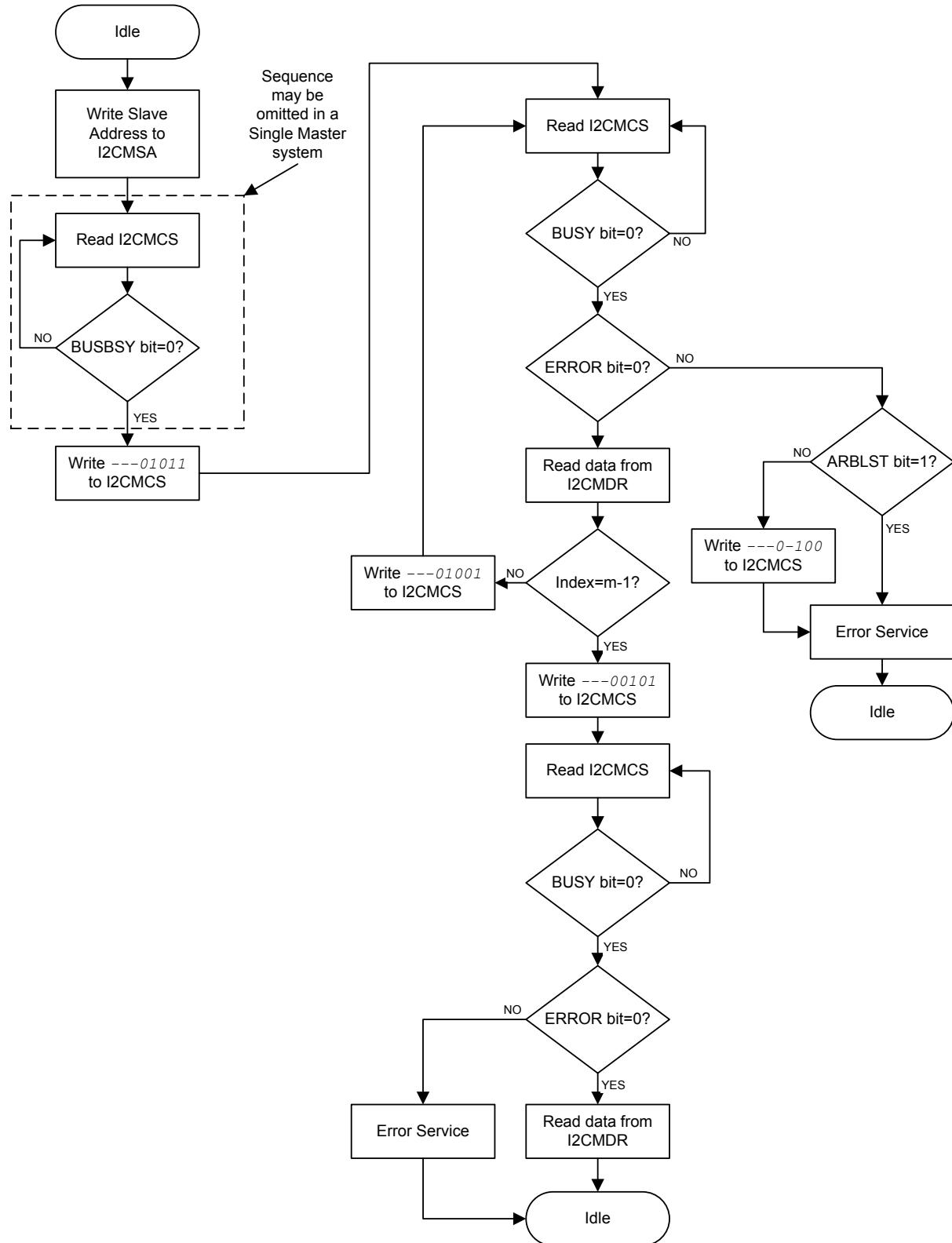
Figure 15-10. Master RECEIVE with Repeated START

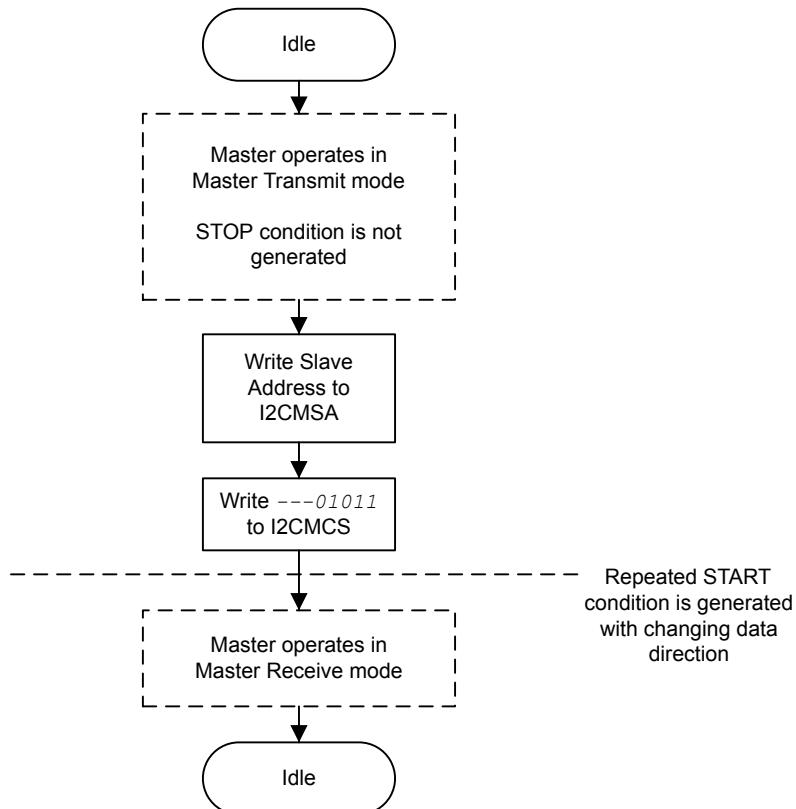
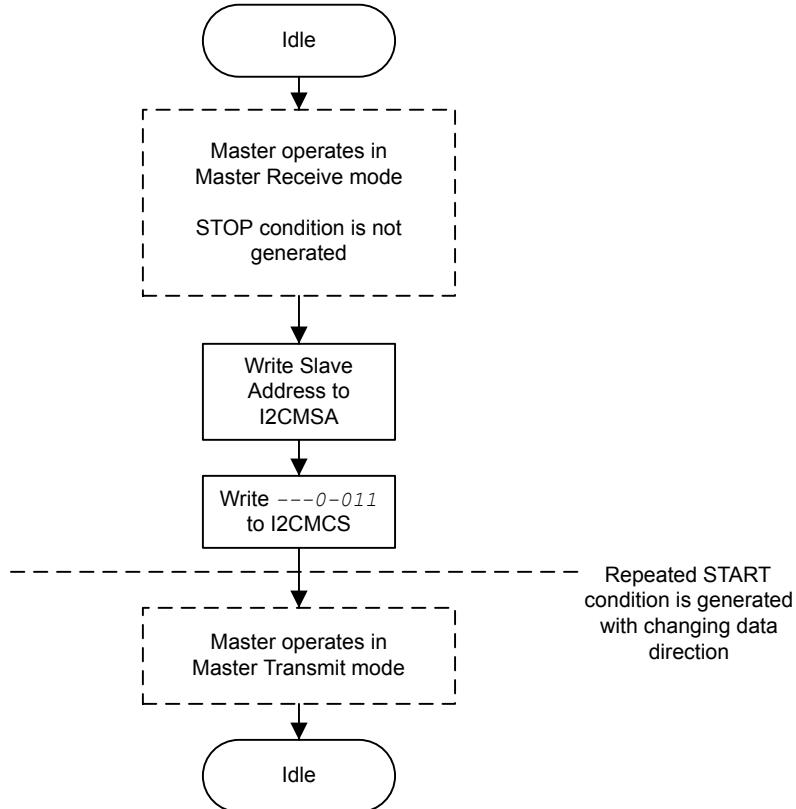
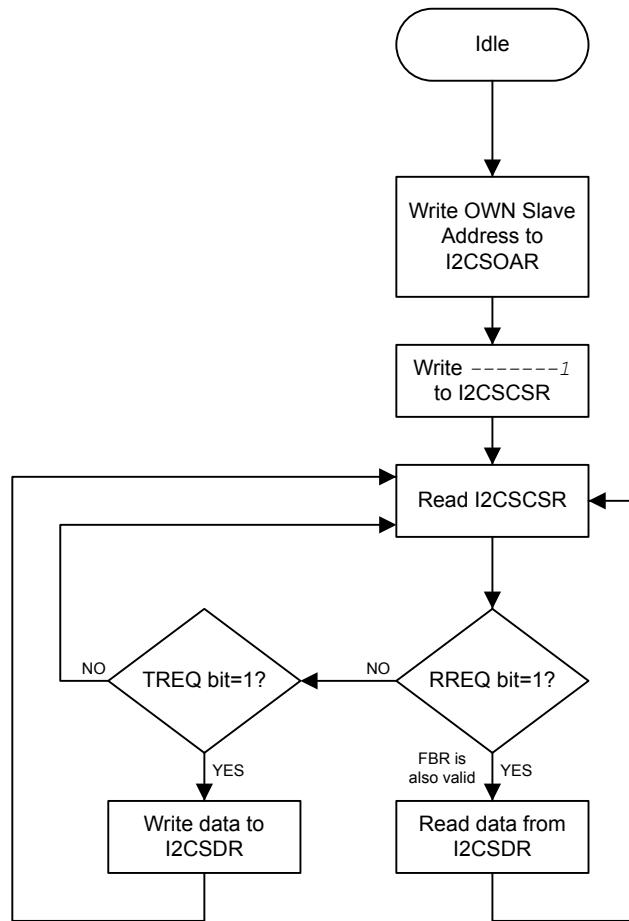
Figure 15-11. Master RECEIVE with Repeated START after TRANSMIT with Repeated START

Figure 15-12. Master TRANSMIT with Repeated START after RECEIVE with Repeated START



15.3.5.2 I²C Slave Command Sequences

Figure 15-13 on page 744 presents the command sequence available for the I²C slave.

Figure 15-13. Slave Command Sequence

15.4 Initialization and Configuration

The following example shows how to configure the I²C module to transmit a single byte as a master. This assumes the system clock is 20 MHz.

1. Enable the I²C clock by writing a value of 0x0000.1000 to the **RCGC1** register in the System Control module (see page 276).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module (see page 285). To find out which GPIO port to enable, refer to Table 24-5 on page 1190.
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register (see page 447). To determine which GPIOs to configure, see Table 24-4 on page 1182.
4. Enable the I²C pins for Open Drain operation. See page 452.
5. Configure the **PMCn** fields in the **GPIOPCTL** register to assign the I²C signals to the appropriate pins. See page 465 and Table 24-5 on page 1190.
6. Initialize the I²C Master by writing the **I2CMCR** register with a value of 0x0000.0010.

- 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.

- Specify the slave address of the master and that the next operation is a Transmit by writing the **I2CMSA** register with a value of 0x0000.0076. This sets the slave address to 0x3B.
- Place data (byte) to be transmitted in the data register by writing the **I2CMDR** register with the desired data.
- Initiate a single byte transmit of the data from Master to Slave by writing the **I2CMCS** register with a value of 0x0000.0007 (STOP, START, RUN).
- Wait until the transmission completes by polling the **I2CMCS** register's **BUSBSY** bit until it has been cleared.
- Check the **ERROR** bit in the **I2CMCS** register to confirm the transmit was acknowledged.

15.5 Register Map

Table 15-4 on page 745 lists the I²C registers. All addresses given are relative to the I²C base address:

- I²C 0: 0x4002.0000
- I²C 1: 0x4002.1000

Note that the I²C module clock must be enabled before the registers can be programmed (see page 276).

The hw_i2c.h file in the StellarisWare® Driver Library uses a base address of 0x800 for the I²C slave registers. Be aware when using registers with offsets between 0x800 and 0x818 that StellarisWare uses an offset between 0x000 and 0x018 with the slave base address.

Table 15-4. Inter-Integrated Circuit (I²C) Interface Register Map

Offset	Name	Type	Reset	Description	See page
I²C Master					
0x000	I2CMSA	R/W	0x0000.0000	I2C Master Slave Address	747
0x004	I2CMCS	R/W	0x0000.0000	I2C Master Control/Status	748
0x008	I2CMDR	R/W	0x0000.0000	I2C Master Data	752
0x00C	I2CMTPR	R/W	0x0000.0001	I2C Master Timer Period	753
0x010	I2CMIMR	R/W	0x0000.0000	I2C Master Interrupt Mask	754
0x014	I2CMRIS	RO	0x0000.0000	I2C Master Raw Interrupt Status	755
0x018	I2CMMIS	RO	0x0000.0000	I2C Master Masked Interrupt Status	756

Table 15-4. Inter-Integrated Circuit (I²C) Interface Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x01C	I2CMICR	WO	0x0000.0000	I2C Master Interrupt Clear	757
0x020	I2CMCR	R/W	0x0000.0000	I2C Master Configuration	758
I²C Slave					
0x800	I2CSOAR	R/W	0x0000.0000	I2C Slave Own Address	759
0x804	I2CSCSR	RO	0x0000.0000	I2C Slave Control/Status	760
0x808	I2CSDR	R/W	0x0000.0000	I2C Slave Data	762
0x80C	I2CSIMR	R/W	0x0000.0000	I2C Slave Interrupt Mask	763
0x810	I2CSRIS	RO	0x0000.0000	I2C Slave Raw Interrupt Status	764
0x814	I2CSMIS	RO	0x0000.0000	I2C Slave Masked Interrupt Status	765
0x818	I2CSICR	WO	0x0000.0000	I2C Slave Interrupt Clear	766

15.6 Register Descriptions (I²C Master)

The remainder of this section lists and describes the I²C master registers, in numerical order by address offset.

Register 1: I²C 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 Transmit (Low).

I2C Master Slave Address (I2CMSA)

I2C 0 base: 0x4002.0000
 I2C 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							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	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.
7:1	SA	R/W	0x00	I ² 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 Transmit (Low).
		Value	Description	
		0	Transmit	
		1	Receive	

Register 2: I²C Master Control/Status (I2CMCS), offset 0x004

This register accesses seven status bits when read and four control bits when written.

The status register consists of seven bits, which when read determine the state of the I²C bus controller.

The control register consists of four bits: the RUN, START, STOP, and ACK bits. The START bit generates 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 repeated START condition. To generate a single transmit cycle, the I²C Master Slave Address (I2CMCSA) register is written with the desired address, the R/S bit is cleared, and the Control register is written with ACK=X (0 or 1), STOP=1, START=1, and RUN=1 to perform the operation and stop. When the operation is completed (or aborted due to an error), the interrupt pin becomes active and the data may be read from the I2CMDR register. When the I²C module operates in Master receiver mode, the ACK bit is normally set causing the I²C bus controller to transmit an acknowledge automatically after each byte. This bit must be cleared when the I²C bus controller requires no further data to be transmitted from the slave transmitter.

Read-Only Status Register

I2C Master Control/Status (I2CMCS)

I2C 0 base: 0x4002.0000

I2C 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
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	BUSY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	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.
6	BUSBSY	RO	0	Bus Busy
		Value	Description	
	0		The I ² C bus is idle.	
	1		The I ² C bus is busy.	
		The bit changes based on the START and STOP conditions.		
5	IDLE	RO	0	I ² C Idle
		Value	Description	
	0		The I ² C controller is not idle.	
	1		The I ² C controller is idle.	

Bit/Field	Name	Type	Reset	Description
4	ARBLST	RO	0	Arbitration Lost Value Description 0 The I ² C controller won arbitration. 1 The I ² C controller lost arbitration.
3	DATACK	RO	0	Acknowledge Data Value Description 0 The transmitted data was acknowledged 1 The transmitted data was not acknowledged.
2	ADRACK	RO	0	Acknowledge Address Value Description 0 The transmitted address was acknowledged 1 The transmitted address was not acknowledged.
1	ERROR	RO	0	Error Value Description 0 No error was detected on the last operation. 1 An error occurred on the last operation. The error can be from the slave address not being acknowledged or the transmit data not being acknowledged.
0	BUSY	RO	0	I ² C Busy Value Description 0 The controller is idle. 1 The controller is busy. When the BUSY bit is set, the other status bits are not valid.

Write-Only Control Register

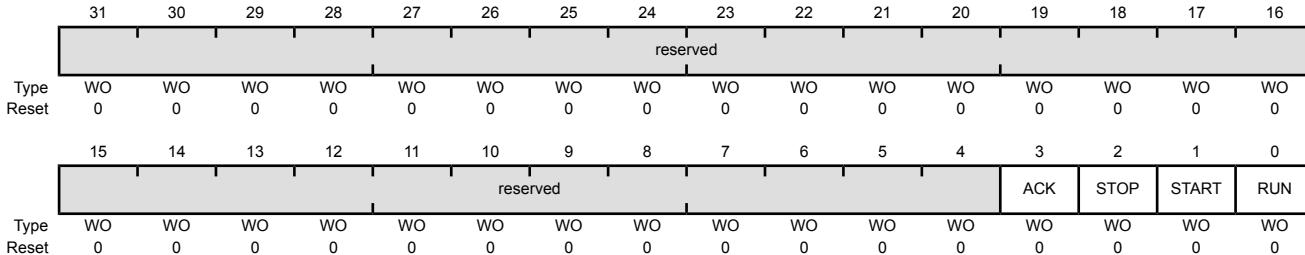
I²C Master Control/Status (I2CMCS)

I²C 0 base: 0x4002.0000

I²C 1 base: 0x4002.1000

Offset 0x004

Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	WO	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	ACK	WO	0	Data Acknowledge Enable Value Description 0 The received data byte is not acknowledged automatically by the master. 1 The received data byte is acknowledged automatically by the master. See field decoding in Table 15-5 on page 750.
2	STOP	WO	0	Generate STOP Value Description 0 The controller does not generate the STOP condition. 1 The controller generates the STOP condition. See field decoding in Table 15-5 on page 750.
1	START	WO	0	Generate START Value Description 0 The controller does not generate the START condition. 1 The controller generates the START or repeated START condition. See field decoding in Table 15-5 on page 750.
0	RUN	WO	0	I ² C Master Enable Value Description 0 The master is disabled. 1 The master is enabled to transmit or receive data. See field decoding in Table 15-5 on page 750.

Table 15-5. Write Field Decoding for I2CMCS[3:0] Field

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
		R/S	ACK	STOP	START	
Idle	0	X ^a	0	1	1	START condition followed by TRANSMIT (master goes to the Master Transmit state).
	0	X	1	1	1	START condition followed by a TRANSMIT and STOP condition (master remains in Idle state).
	1	0	0	1	1	START condition followed by RECEIVE operation with negative ACK (master goes to the Master Receive state).
	1	0	1	1	1	START condition followed by RECEIVE and STOP condition (master remains in Idle state).
	1	1	0	1	1	START condition followed by RECEIVE (master goes to the Master Receive state).
	1	1	1	1	1	Illegal
	All other combinations not listed are non-operations.				NOP	

Table 15-5. Write Field Decoding for I2CMCS[3:0] Field (continued)

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
		R/S	ACK	STOP	START	
Master Transmit	X	X	0	0	1	TRANSMIT operation (master remains in Master Transmit state).
	X	X	1	0	0	STOP condition (master goes to Idle state).
	X	X	1	0	1	TRANSMIT followed by STOP condition (master goes to Idle state).
	0	X	0	1	1	Repeated START condition followed by a TRANSMIT (master remains in Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by TRANSMIT 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 TRANSMIT and STOP condition (master goes to Idle state).
	1	1	0	1	1	Repeated START condition followed by RECEIVE (master goes to Master Receive state).
	1	1	1	1	1	Illegal.
	All other combinations not listed are non-operations.					NOP.
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). ^b
	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 TRANSMIT (master goes to Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by TRANSMIT 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: I²C Master Data (I2CMDR), offset 0x008

Important: Use caution when reading this register. Performing a read may change bit status.

This register contains the data to be transmitted when in the Master Transmit state and the data received when in the Master Receive state.

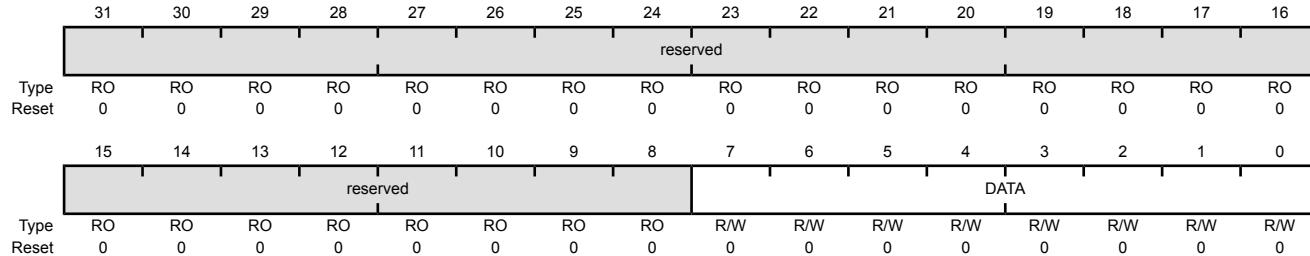
I2C Master Data (I2CMDR)

I2C 0 base: 0x4002.0000

I2C 1 base: 0x4002.1000

Offset 0x008

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	Data Transferred Data transferred during transaction.

Register 4: I²C Master Timer Period (I2CMTPR), offset 0x00C

This register specifies the period of the SCL clock.

Caution – Take care not to set bit 7 when accessing this register as unpredictable behavior can occur.

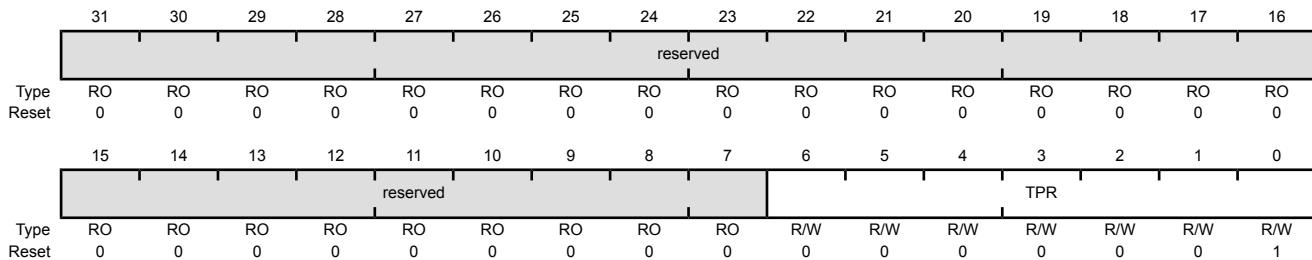
I2C Master Timer Period (I2CMTPR)

I2C 0 base: 0x4002.0000

I2C 1 base: 0x4002.1000

Offset 0x00C

Type R/W, reset 0x0000.0001



Bit/Field	Name	Type	Reset	Description
31:7	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.
6:0	TPR	R/W	0x1	<p>SCL Clock Period This field specifies the period of the SCL clock. $SCL_PRD = 2 \times (1 + TPR) \times (SCL_LP + SCL_HP) \times CLK_PRD$ where: SCL_PRD is the SCL line period (I²C clock). TPR is the Timer Period register value (range of 1 to 127). SCL_LP is the SCL Low period (fixed at 6). SCL_HP is the SCL High period (fixed at 4). CLK_PRD is the system clock period in ns.</p>

Register 5: I²C Master Interrupt Mask (I2CMIMR), offset 0x010

This register controls whether a raw interrupt is promoted to a controller interrupt.

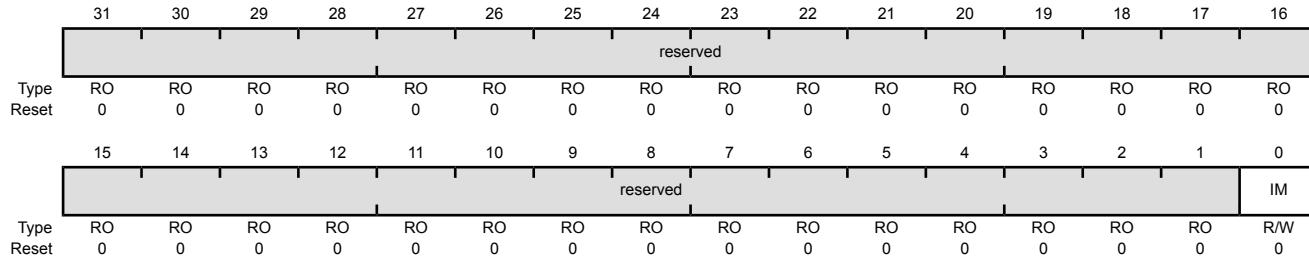
I2C Master Interrupt Mask (I2CMIMR)

I2C 0 base: 0x4002.0000

I2C 1 base: 0x4002.1000

Offset 0x010

Type R/W, reset 0x0000.0000



Bit/Field Name Type Reset Description

31:1 reserved RO 0x0000.0000 Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

0 IM R/W 0 Interrupt Mask

Value	Description
1	The master interrupt is sent to the interrupt controller when the RIS bit in the I2CMRIS register is set.
0	The RIS interrupt is suppressed and not sent to the interrupt controller.

Register 6: I²C Master Raw Interrupt Status (I2CMRIS), offset 0x014

This register specifies whether an interrupt is pending.

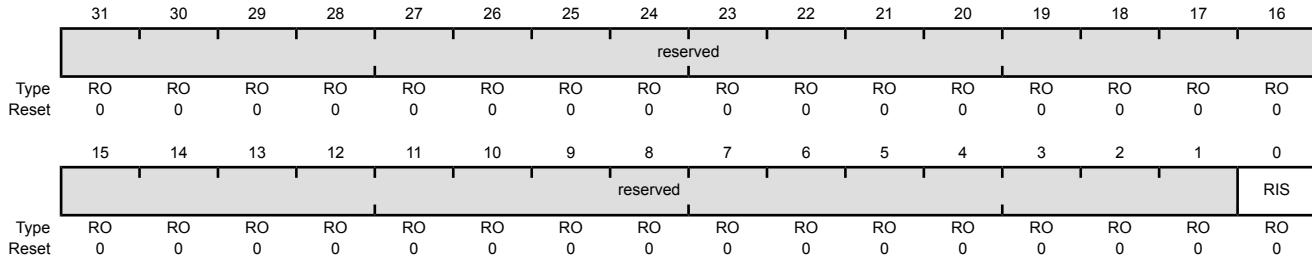
I2C Master Raw Interrupt Status (I2CMRIS)

I2C 0 base: 0x4002.0000

I2C 1 base: 0x4002.1000

Offset 0x014

Type RO, reset 0x0000.0000



Bit/Field Name Type Reset Description

31:1 reserved RO 0x0000.0000 Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

0 RIS RO 0 Raw Interrupt Status

Value	Description
1	A master interrupt is pending.
0	No interrupt.

This bit is cleared by writing a 1 to the IC bit in the I2CMICR register.

Register 7: I²C Master Masked Interrupt Status (I2CMMIS), offset 0x018

This register specifies whether an interrupt was signaled.

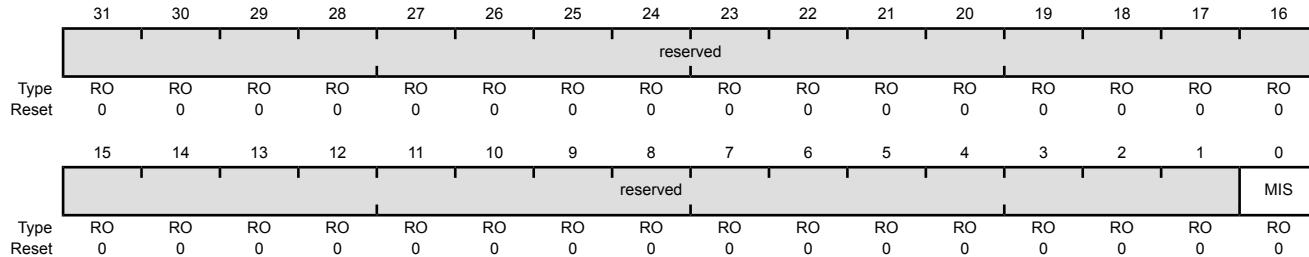
I2C Master Masked Interrupt Status (I2CMMIS)

I2C 0 base: 0x4002.0000

I2C 1 base: 0x4002.1000

Offset 0x018

Type RO, reset 0x0000.0000



Bit/Field Name Type Reset Description

31:1 reserved RO 0x0000.0000 Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

0 MIS RO 0 Masked Interrupt Status

Value Description

1 An unmasked master interrupt was signaled and is pending.

0 An interrupt has not occurred or is masked.

This bit is cleared by writing a 1 to the IC bit in the I2CMICR register.

Register 8: I²C Master Interrupt Clear (I2CMICR), offset 0x01C

This register clears the raw interrupt.

I2C Master Interrupt Clear (I2CMICR)

I2C 0 base: 0x4002.0000

I2C 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	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IC	WO	0	Interrupt Clear Writing a 1 to this bit clears the RIS bit in the I2CMRIS register and the MIS bit in the I2CMMIS register. A read of this register returns no meaningful data.

Register 9: I²C Master Configuration (I2CMCR), offset 0x020

This register configures the mode (Master or Slave) and sets the interface for test mode loopback.

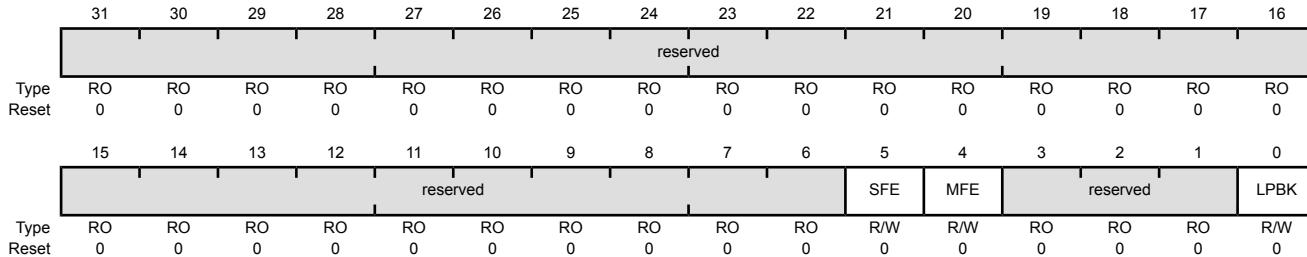
I2C Master Configuration (I2CMCR)

I2C 0 base: 0x4002.0000

I2C 1 base: 0x4002.1000

Offset 0x020

Type R/W, reset 0x0000.0000



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	SFE	R/W	0	I ² C Slave Function Enable
		Value	Description	
		1	Slave mode is enabled.	
		0	Slave mode is disabled.	
4	MFE	R/W	0	I ² C Master Function Enable
		Value	Description	
		1	Master mode is enabled.	
		0	Master mode is disabled.	
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	LPBK	R/W	0	I ² C Loopback
		Value	Description	
		1	The controller in a test mode loopback configuration.	
		0	Normal operation.	

15.7 Register Descriptions (I²C Slave)

The remainder of this section lists and describes the I²C slave registers, in numerical order by address offset.

Register 10: I²C Slave Own Address (I2CSOAR), offset 0x800

This register consists of seven address bits that identify the Stellaris I²C device on the I²C bus.

I2C Slave Own Address (I2CSOAR)

I2C 0 base: 0x4002.0000

I2C 1 base: 0x4002.1000

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															OAR
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	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.
6:0	OAR	R/W	0x00	I ² C Slave Own Address This field specifies bits A6 through A0 of the slave address.

Register 11: I²C Slave Control/Status (I2CSCCSR), offset 0x804

This register accesses one control bit when written, and three status bits when read.

The read-only Status register consists of three bits: the FBR, RREQ, and TREQ bits. The First Byte Received (FBR) bit is set only after the Stellaris device detects its own slave address and receives the first data byte from the I²C master. The Receive Request (RREQ) bit indicates that the Stellaris I²C device has received a data byte from an I²C master. Read one data byte from the **I²C Slave Data (I2CSDR)** register to clear the RREQ bit. The Transmit Request (TREQ) bit indicates that the Stellaris I²C device is addressed as a Slave Transmitter. Write one data byte into the **I²C 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 I²C slave operation.

Read-Only Status Register

I²C Slave Control/Status (I2CSCCSR)

I²C 0 base: 0x4002.0000

I²C 1 base: 0x4002.1000

Offset 0x804

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	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	FBR	RO	0	First Byte Received
	Value	Description		
	1	The first byte following the slave's own address has been received.		
	0	The first byte has not been received.		
	This bit is only valid when the RREQ bit is set and is automatically cleared when data has been read from the I2CSDR register.			
	Note: This bit is not used for slave transmit operations.			
1	TREQ	RO	0	Transmit Request
	Value	Description		
	1	The I ² C controller has been addressed as a slave transmitter and is using clock stretching to delay the master until data has been written to the I2CSDR register.		
	0	No outstanding transmit request.		

Bit/Field	Name	Type	Reset	Description
0	RREQ	RO	0	Receive Request
Value Description				
1				The I ² C controller has outstanding receive data from the I ² C master and is using clock stretching to delay the master until the data has been read from the I2CSDR register.
0				No outstanding receive data.

Write-Only Control Register

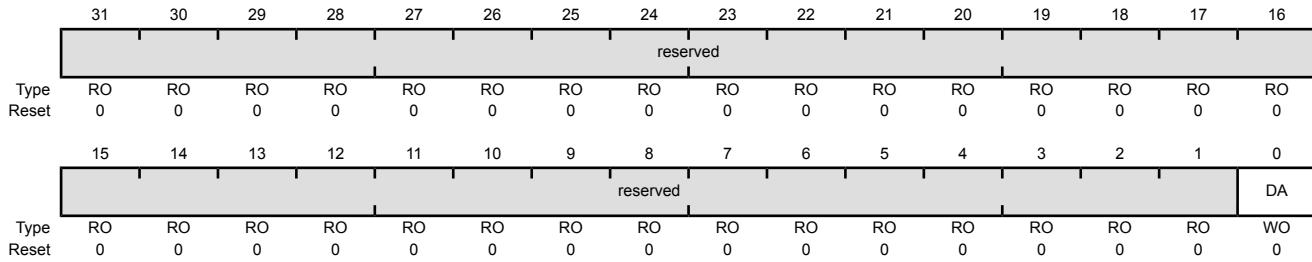
I2C Slave Control/Status (I2CSCCSR)

I2C 0 base: 0x4002.0000

I2C 1 base: 0x4002.1000

Offset 0x804

Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	DA	WO	0	Device Active
Value Description				
0				Disables the I ² C slave operation.
1				Enables the I ² C slave operation.

Register 12: I²C Slave Data (I2CSDR), offset 0x808

Important: Use caution when reading this register. Performing a read may change bit status.

This register contains the data to be transmitted when in the Slave Transmit state, and the data received when in the Slave Receive state.

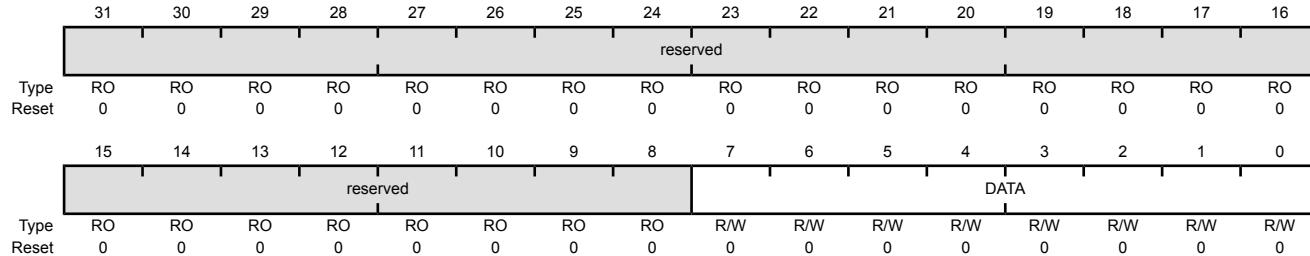
I2C Slave Data (I2CSDR)

I2C 0 base: 0x4002.0000

I2C 1 base: 0x4002.1000

Offset 0x808

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 Data for Transfer
This field contains the data for transfer during a slave receive or transmit operation.

Register 13: I²C Slave Interrupt Mask (I2CSIMR), offset 0x80C

This register controls whether a raw interrupt is promoted to a controller interrupt.

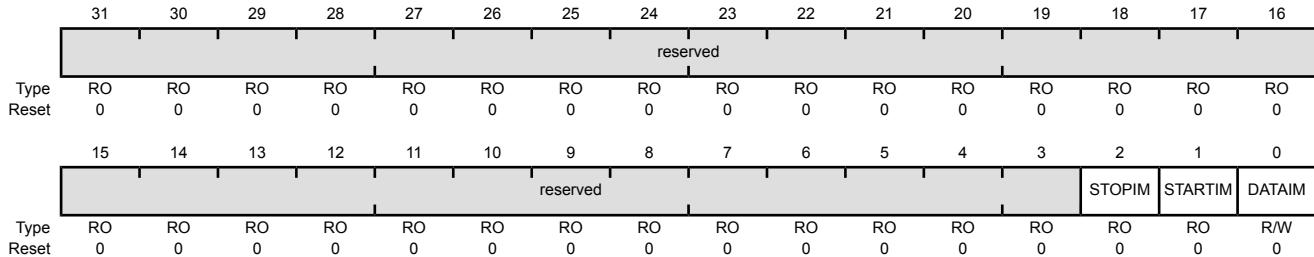
I2C Slave Interrupt Mask (I2CSIMR)

I2C 0 base: 0x4002.0000

I2C 1 base: 0x4002.1000

Offset 0x80C

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPIM	RO	0	Stop Condition Interrupt Mask
	Value	Description		
	1	The STOP condition interrupt is sent to the interrupt controller when the STOPRIS bit in the I2CSRIS register is set.		
	0	The STOPRIS interrupt is suppressed and not sent to the interrupt controller.		
1	STARTIM	RO	0	Start Condition Interrupt Mask
	Value	Description		
	1	The START condition interrupt is sent to the interrupt controller when the STARTRIS bit in the I2CSRIS register is set.		
	0	The STARTRIS interrupt is suppressed and not sent to the interrupt controller.		
0	DATAIM	R/W	0	Data Interrupt Mask
	Value	Description		
	1	The data received or data requested interrupt is sent to the interrupt controller when the DATARIS bit in the I2CSRIS register is set.		
	0	The DATARIS interrupt is suppressed and not sent to the interrupt controller.		

Register 14: I²C Slave Raw Interrupt Status (I2CSRIS), offset 0x810

This register specifies whether an interrupt is pending.

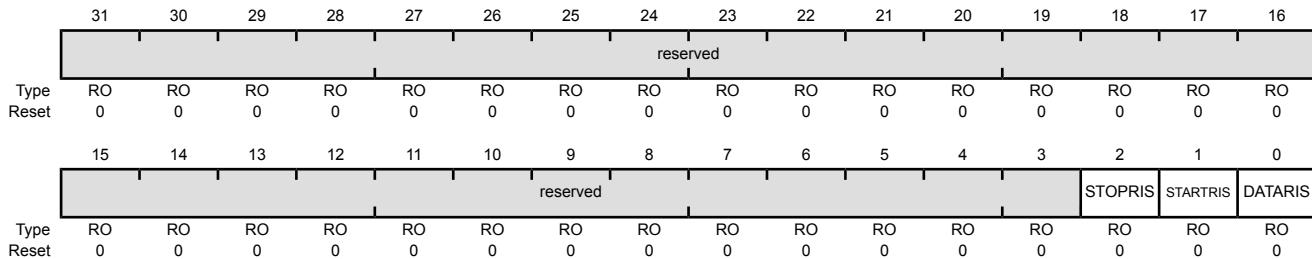
I2C Slave Raw Interrupt Status (I2CSRIS)

I2C 0 base: 0x4002.0000

I2C 1 base: 0x4002.1000

Offset 0x810

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPRIS	RO	0	Stop Condition Raw Interrupt Status
	Value Description			
	1	A STOP condition interrupt is pending.		
	0	No interrupt.		
	This bit is cleared by writing a 1 to the STOPIC bit in the I2CSICR register.			
1	STARTRIS	RO	0	Start Condition Raw Interrupt Status
	Value Description			
	1	A START condition interrupt is pending.		
	0	No interrupt.		
	This bit is cleared by writing a 1 to the STARTIC bit in the I2CSICR register.			
0	DATARIS	RO	0	Data Raw Interrupt Status
	Value Description			
	1	A data received or data requested interrupt is pending.		
	0	No interrupt.		
	This bit is cleared by writing a 1 to the DATAIC bit in the I2CSICR register.			

Register 15: I²C Slave Masked Interrupt Status (I2CSMIS), offset 0x814

This register specifies whether an interrupt was signaled.

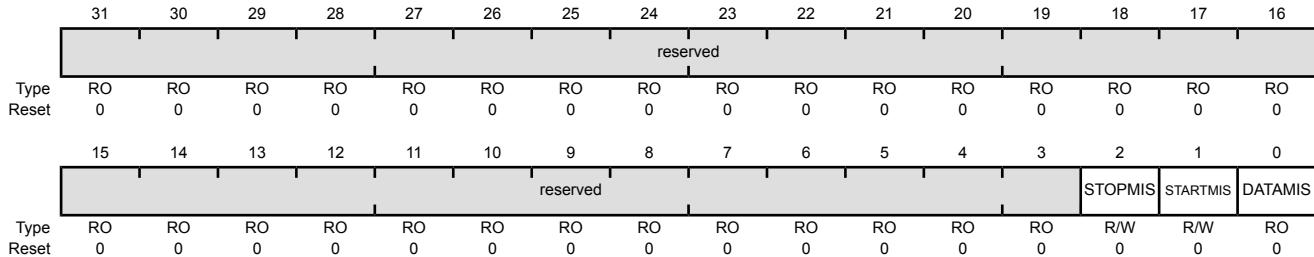
I2C Slave Masked Interrupt Status (I2CSMIS)

I2C 0 base: 0x4002.0000

I2C 1 base: 0x4002.1000

Offset 0x814

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPMIS	R/W	0	Stop Condition Masked Interrupt Status Value Description 1 An unmasked STOP condition interrupt was signaled is pending. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the STOPIC bit in the I2CSICR register.
1	STARTMIS	R/W	0	Start Condition Masked Interrupt Status Value Description 1 An unmasked START condition interrupt was signaled is pending. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the STARTIC bit in the I2CSICR register.
0	DATAMIS	RO	0	Data Masked Interrupt Status Value Description 1 An unmasked data received or data requested interrupt was signaled is pending. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the DATAIC bit in the I2CSICR register.

Register 16: I²C Slave Interrupt Clear (I2CSICR), offset 0x818

This register clears the raw interrupt. A read of this register returns no meaningful data.

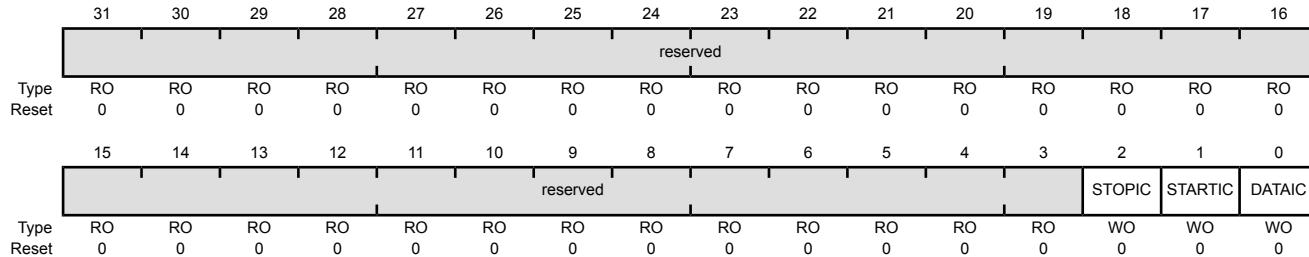
I²C Slave Interrupt Clear (I2CSICR)

I²C 0 base: 0x4002.0000

I²C 1 base: 0x4002.1000

Offset 0x818

Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPIC	WO	0	Stop Condition Interrupt Clear Writing a 1 to this bit clears the STOPRIS bit in the I2CSRIS register and the STOPMIS bit in the I2CSMIS register. A read of this register returns no meaningful data.
1	STARTIC	WO	0	Start Condition Interrupt Clear Writing a 1 to this bit clears the STOPRIS bit in the I2CSRIS register and the STOPMIS bit in the I2CSMIS register. A read of this register returns no meaningful data.
0	DATAIC	WO	0	Data Interrupt Clear Writing a 1 to this bit clears the STOPRIS bit in the I2CSRIS register and the STOPMIS bit in the I2CSMIS register. A read of this register returns no meaningful data.

16 Inter-Integrated Circuit Sound (I²S) Interface

The I²S module is a configurable serial audio core that contains a transmit module and a receive module. The module is configurable for the I²S 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.

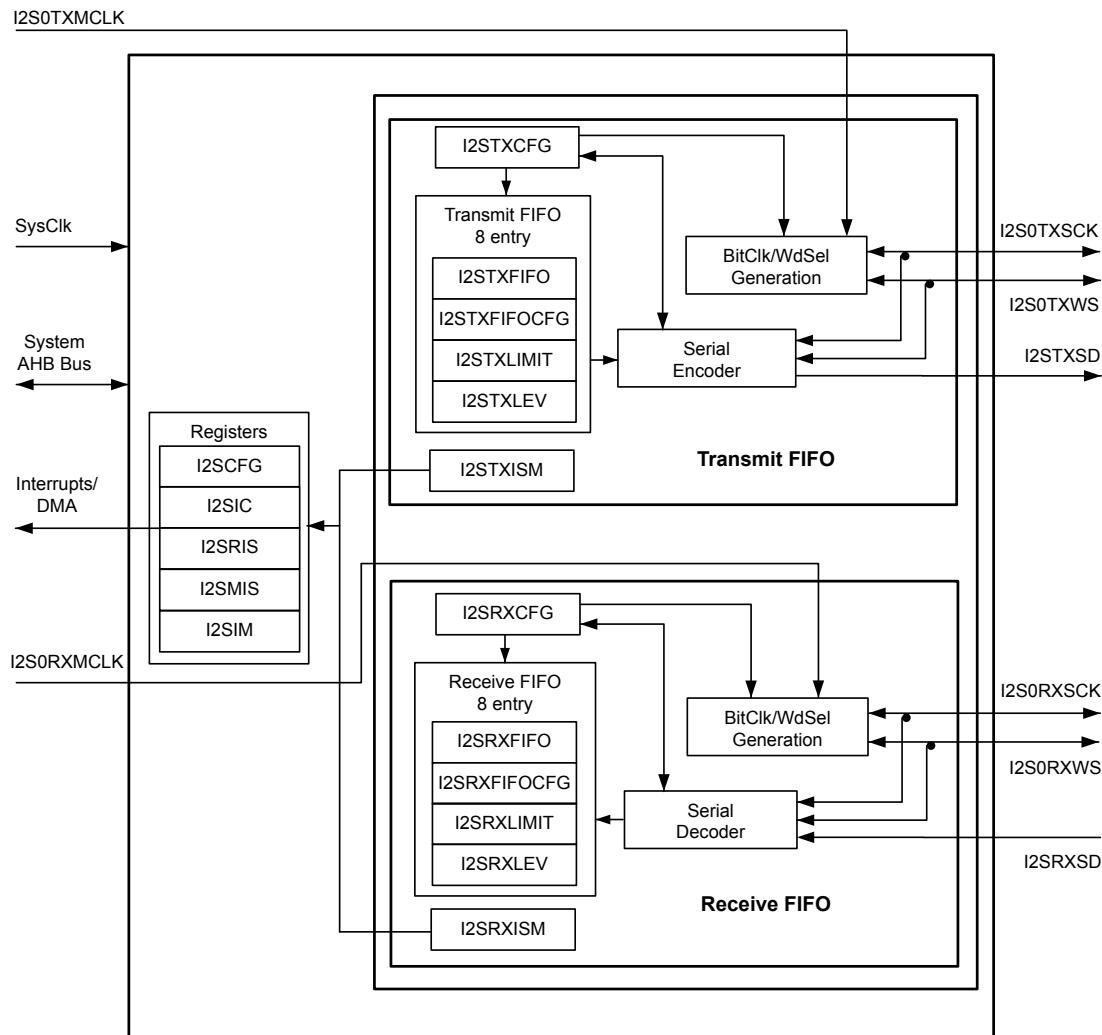
Both the transmitter and receiver are capable of being a master or a slave.

The Stellaris® I²S module has the following features:

- Configurable audio format supporting I²S, 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 μDMA 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 (μDMA)
 - Separate channels for transmit and receive
 - Burst requests
 - Channel requests asserted when FIFO contains required amount of data

16.1 Block Diagram

Figure 16-1. I²S Block Diagram



16.2 Signal Description

Table 16-1 on page 769 and Table 16-2 on page 769 list the external signals of the I²S module and describe the function of each. The I²S module signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the I²S signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 447) should be set to choose the I²S function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 465) to assign the I²S signal to the specified GPIO port pin. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 423.

Table 16-1. Signals for I²S (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
I ² S0RXMCLK	29 98	PA3 (9) PD5 (8)	I/O	TTL	I ² S module 0 receive master clock.
I ² S0RXSCK	10	PD0 (8)	I/O	TTL	I ² S module 0 receive clock.
I ² S0RXSD	28 97	PA2 (9) PD4 (8)	I/O	TTL	I ² S module 0 receive data.
I ² S0RXWS	11	PD1 (8)	I/O	TTL	I ² S module 0 receive word select.
I ² S0TXMCLK	61	PF1 (8)	I/O	TTL	I ² S module 0 transmit master clock.
I ² S0TXSCK	30 90 99	PA4 (9) PB6 (9) PD6 (8)	I/O	TTL	I ² S module 0 transmit clock.
I ² S0TXSD	5 47	PE5 (9) PF0 (8)	I/O	TTL	I ² S module 0 transmit data.
I ² S0TXWS	6 31 100	PE4 (9) PA5 (9) PD7 (8)	I/O	TTL	I ² S module 0 transmit word select.

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

Table 16-2. Signals for I²S (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
I ² S0RXMCLK	L4 C6	PA3 (9) PD5 (8)	I/O	TTL	I ² S module 0 receive master clock.
I ² S0RXSCK	G1	PD0 (8)	I/O	TTL	I ² S module 0 receive clock.
I ² S0RXSD	M4 B5	PA2 (9) PD4 (8)	I/O	TTL	I ² S module 0 receive data.
I ² S0RXWS	G2	PD1 (8)	I/O	TTL	I ² S module 0 receive word select.
I ² S0TXMCLK	H12	PF1 (8)	I/O	TTL	I ² S module 0 transmit master clock.
I ² S0TXSCK	L5 A7 A3	PA4 (9) PB6 (9) PD6 (8)	I/O	TTL	I ² S module 0 transmit clock.
I ² S0TXSD	B3 M9	PE5 (9) PF0 (8)	I/O	TTL	I ² S module 0 transmit data.
I ² S0TXWS	B2 M5 A2	PE4 (9) PA5 (9) PD7 (8)	I/O	TTL	I ² S module 0 transmit word select.

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

16.3 Functional Description

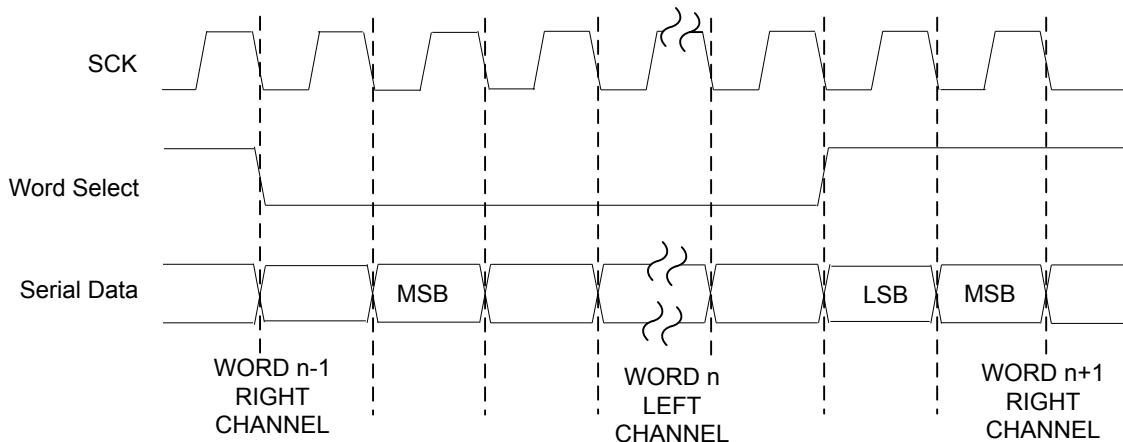
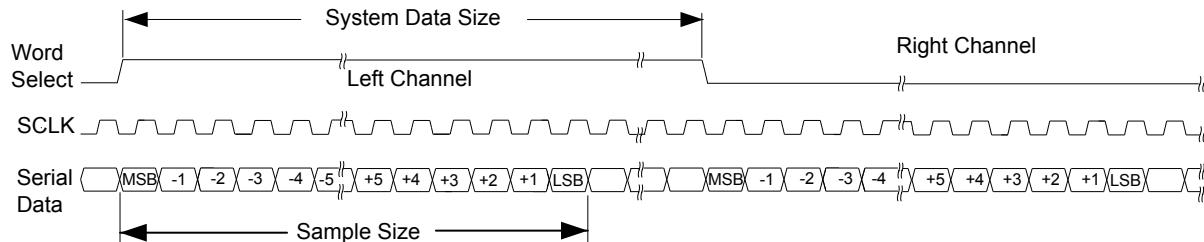
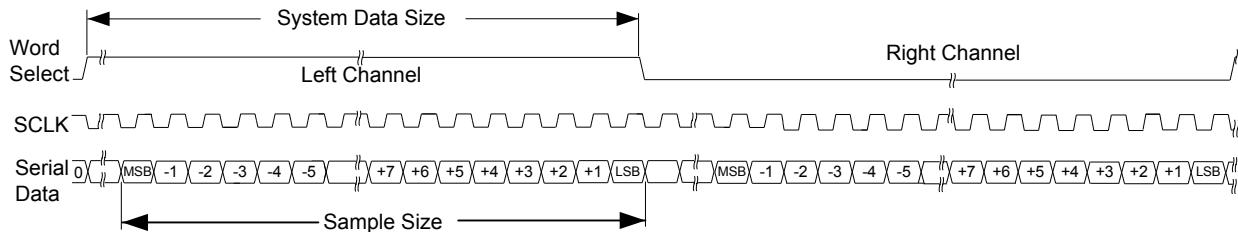
The Inter-Integrated Circuit Sound (I²S) module contains separate transmit and receive engines. Each engine consists of the following:

- 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 I²S block is as follows:

- Configuration
 - Overall I²S module configuration in the **I²S 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 **I²S Transmit Module Configuration (I2STXCFG)** and **I²S 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 **I²S Transmit FIFO Configuration (I2STXFIFO CFG)** and **I²S Receive FIFO Configuration (I2SRXFIFO CFG)** 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 **I²S Transmit FIFO Data (I2STXFIFO)** and **I²S Receive FIFO Data (I2SRXFIFO)** registers
 - Information on FIFO data levels in the **I²S Transmit FIFO Level (I2STXLEV)** and **I²S Receive FIFO Level (I2SRXLEV)** registers
 - Configuration for FIFO service requests based on FIFO levels in the **I²S Transmit FIFO Limit (I2STXLIMIT)** and **I²S Receive FIFO Limit (I2SRXLIM)** registers
- Interrupt Control
 - Interrupt masking configuration in the **I²S Interrupt Mask (I2SIM)** register
 - Raw and masked interrupt status in the **I²S Raw Interrupt Status (I2SRIS)** and **I²S Masked Interrupt Status (I2SMIS)** registers
 - Interrupt clearing through the **I²S Interrupt Clear (I2SIC)** register
 - Configuration for FIFO service requests interrupts and transmit/receive error interrupts in the **I²S Transmit Interrupt Status and Mask (I2STXISM)** and **I²S Receive Interrupt Status and Mask (I2SRXISM)** registers

Figure 16-2 on page 771 provides an example of an I²S data transfer. Figure 16-3 on page 771 provides an example of an Left-Justified data transfer. Figure 16-4 on page 771 provides an example of an Right-Justified data transfer.

Figure 16-2. I²S Data Transfer**Figure 16-3. Left-Justified Data Transfer****Figure 16-4. Right-Justified Data Transfer**

16.3.1 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.

16.3.1.1 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 I²S, 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 **I²S 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 **I²S Module Configuration (I2SCFG)** register.

16.3.1.2 FIFO Operation

The transmit FIFO stores eight Mono samples or eight Stereo sample-pairs of data and is accessed through the **I²S Transmit FIFO Data (I2STXFIFO)** register. The FIFO interface for the audio data is different based on the Write mode, defined by the **I²S Transmit FIFO Configuration (I2STXFIFO CFG)** Compact Stereo Sample Size bit (CSS) and the **I2STXCFG** Write Mode field (WM). All data samples are MSB-aligned. Table 16-3 on page 772 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 **I2STXFIFO CFG** register.

Table 16-3. I²S Transmit FIFO Interface

WM field in I2STXCFG	CSS bit in I2STXFIFO CFG	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 **I²S 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.

16.3.1.3 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 243 for the setup for the master I2S0TXMCLK source. An external transmit I2S0TXMCLK does not require the use of the PLL and is selected using the TXSLV bit in the **I2SCFG** register.

The following tables show combinations of the TXINT and TXFRAC bits in the **I²S 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:

$$\text{MCLK} = \text{Fs} \times 256 \quad \text{PLL} = 400 \text{ MHz}$$

The Integer value is taken from the result of the following calculation:

$$\text{ROUND}(\text{PLL}/\text{MCLK})$$

The remaining fractional component is converted to binary, and the first four bits are the Fractional value.

Table 16-4. Crystal Frequency (Values from 3.5795 MHz to 5 MHz)

Sampling Frequency Fs (kHz)	Crystal Frequency (MHz)											
	3.5795		3.6864		4		4.096		4.9152		5	
	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional	Integer	Fractional
8	195	12	194	6	195	5	196	0	194	6	195	5
11.025	142	1	141	1	141	12	142	4	141	1	141	12
12	130	8	129	10	130	3	130	11	129	10	130	3
16	97	14	97	3	97	10	98	0	97	3	97	10
22.05	71	0	70	8	70	14	71	2	70	8	70	14
24	65	4	64	13	65	2	65	5	64	13	65	2
32	48	15	48	10	48	13	49	0	48	10	48	13
44.1	35	8	35	4	35	7	35	9	35	4	35	7
48	32	10	32	6	32	9	32	11	32	6	32	9
64	24	8	24	5	24	7	24	8	24	5	24	7
88.2	17	12	17	10	17	11	17	12	17	10	17	11
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	14	8	13	8	14	8	14	8	13	8	14
192	Not supported		Not supported		8	2	8	3	Not supported		8	2

Table 16-5. Crystal Frequency (Values from 5.12 MHz to 8.192 MHz)

Sampling Frequency 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	11
11.025	141	8	141	12	141	8	141	1	141	12	141	4
12	130	0	130	3	130	0	129	10	130	3	129	12
16	97	8	97	10	97	8	97	3	97	10	97	5
22.05	70	12	70	14	70	12	70	8	70	14	70	10
24	65	0	65	2	65	0	64	13	65	2	64	14
32	48	12	48	13	48	12	48	10	48	13	48	11
44.1	35	6	35	7	35	6	35	4	35	7	35	5
48	32	8	32	9	32	8	32	6	32	9	32	7
64	24	6	24	7	24	6	24	5	24	7	24	5
88.2	17	11	17	11	17	11	17	10	17	11	17	11
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	14	Not supported		8	13	8	14	8	13
192	8	2	8	2	8	2	Not supported		8	2	8	2

Table 16-6. Crystal Frequency (Values from 10 MHz to 14.3181 MHz)

Sampling Frequency 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	3	195	12
11.025	141	12	141	12	142	4	140	15	142	1
12	130	3	130	3	130	11	129	8	130	8
16	97	10	97	10	98	0	97	2	97	14
22.05	70	14	70	14	71	2	70f	7	71	0
24	65	2	65	2	65	5	64	12	65	4
32	48	13	48	13	49	0	48	9	48	15
44.1	35	7	35	7	35	9	35	4	35	8
48	32	9	32	9	32	11	32	6	32	10
64	24	7	24	7	24	8	24	4	24	8
88.2	17	11	17	11	17	12	17	10	17	12
96	16	4	16	4	16	5	16	3	16	5
128	12	3	12	3	12	4	12	2	12	4
176.4	8	14	8	14	8	14	8	13	8	14
192	8	2	8	2	8	3	Not supported		Not supported	

Table 16-7. Crystal Frequency (Values from 16 MHz to 16.384 MHz)

Sampling Frequency Fs (kHz)	Crystal Frequency (MHz)			
	16		16.384	
	Integer	Fractional	Integer	Fractional
8	195	5	192	0
11.025	141	12	139	5
12	130	3	128	0
16	97	10	96	0
22.05	70	14	69	11
24	65	2	64	0
32	48	13	48	0
44.1	35	7	34	13
48	32	9	32	0
64	24	7	24	0
88.2	17	11	17	7
96	16	4	16	0
128	12	3	12	0
176.4	8	14	8	11
192	8	2	8	0

16.3.1.4 Interrupt Control

A single 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 TXSRIM and TXWEIM bits in the I²S Interrupt Mask (I2SIM)

register. The status of the interrupt source is indicated by the **I²S Raw Interrupt Status (I2SRIS)** register. The status of enabled interrupts is indicated by the **I²S Masked Interrupt Status (I2SMIS)** register. The FIFO level interrupt has a second level of masking using the FFM bit in the **I²S 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 **I²S Transmit FIFO Level (I2STXLEV)** register) is below the FIFO limit (programmed using the **I²S Transmit FIFO Limit (I2STXLIMIT)** register) and both the TXSRIM and FFM bits are set. If software attempts to write to a full FIFO, a Transmit FIFO Write error occurs (indicated by the TXWERIS bit in the **I²S Raw Interrupt Status (I2SRIS)** register). The TXWERIS bit in the **I2SRIS** register and the TXWEMIS bit in the **I2SMIS** register are cleared by setting the TXWEIC bit in the **I²S Interrupt Clear (I2SIC)** register.

16.3.1.5 DMA Support

The μDMA can be used to more efficiently stream data to and from the I²S bus. The I²S transmit and receive modules have separate μDMA channels. The FIFO Interrupt Mask bit (FFM) in the **I2STXISM** register must be set for the request signaling to propagate to the μDMA module. See “Micro Direct Memory Access (μDMA)” on page 365 for channel configuration.

The I²S module uses the μDMA burst request signal, not the single request. Thus each time a μDMA request is made, the μDMA controller transfers the number of items specified as the burst size for the μDMA channel. Therefore, the μDMA channel burst size and the I²S FIFO service request limit must be set to the same value (using the LIMIT field in the **I2STXLIMIT** register).

16.3.2 Receive

The receiver consists of a serial decoder, an 8-entry FIFO, and control logic. The receiver has independent MCLK (I2S0RXMCLK), SCLK (I2S0RXSCK), and Word-Select (I2S0RXWS) signals.

16.3.2.1 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 I²S, 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 **I²S 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 I2S0RXSCK 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.

16.3.2.2 FIFO Operation

The receive FIFO stores eight Mono samples or eight Stereo sample-pairs of data and is accessed through the **I²S Receive FIFO Data (I2SRXFIFO)** register. Table 16-8 on page 776 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 **I²S Receive FIFO Configuration (I2SRXFIFO CFG)** register. This enables reads from a single channel, where the channel selected can be either the right or left as determined by the LRP bit in the **I2SRXCFG** register.

Table 16-8. I²S Receive FIFO Interface

RM bit in I2RXCFG	css bit in I2SRXFIFO CFG	Read Mode	Sample Width	Samples per FIFO Read	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 I2SRXFIFO CFG register must be set.)	8-32 bits	1	MSB

The number of samples in the receive FIFO can be read using the **I²S 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.

16.3.2.3 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 243 for the setup for the master I2S0RXMCLK source. An external transmit I2S0RXMCLK does not require the use of the PLL and is selected using the **RXSLV** bit in the **I2SCFG** register.

Refer to “Clock Control” on page 772 for combinations of the **RXINT** and **RXFrac** bits in the **I²S 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%.

16.3.2.4 Interrupt Control

A single 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 **RXSRIIM** and **RXREIIM** 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 **FFM** bit in the **I²S Receive Interrupt Status and Mask (I2SRXISM)** 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 **I²S Receive FIFO Level (I2SRXLEV)** register) is above the FIFO limit (programmed using the **I²S Receive FIFO Limit (I2SRXLIMIT)** register) and both the **RXSRIIM** and **FFM** 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 **RXRERIS** bit in the **I2SRIS** register). The **RXRERIS** bit in the **I2SRIS** register and the **RXREMIS** bit in the **I2SMIS** register are cleared by setting the **RXREIC** bit in the **I2SIC** register.

16.3.2.5 DMA Support

The μDMA can be used to more efficiently stream data to and from the I²S bus. The I²S transmit and receive modules have separate μDMA channels. The FIFO Interrupt Mask bit (FFM) in the **I2SRXISM** register must be set for the request signaling to propagate to the μDMA module. See “Micro Direct Memory Access (μDMA)” on page 365 for channel configuration.

The I²S module uses the μDMA burst request signal, not the single request. Thus each time a μDMA request is made, the μDMA controller transfers the number of items specified as the burst size for the μDMA channel. Therefore, the μDMA channel burst size and the I²S FIFO service request limit must be set to the same value (using the LIMIT field in the **I2SRXLIMIT** register).

16.4 Initialization and Configuration

The default setup for the I²S transmit and receive is to use external MCLK, external SCLK, Stereo, I²S 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 I²S peripheral clock by writing a value of 0x1000.0000 to the **RCGC1** register in the System Control module (see page 276).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module (see page 285). To find out which GPIO port to enable, refer to Table 24-5 on page 1190.
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register (see page 447). To determine which GPIOs to configure, see Table 24-4 on page 1182.
4. Configure the **PMCn** fields in the **GPIOPCTL** register to assign the I²S signals to the appropriate pins (see page 465 and Table 24-5 on page 1190).
5. 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 226).
 - Set the MCLK dividers and enable them by writing 0x0208.0208 to the **I2SMCLKCFG** register in the System Control module (see page 243).
 - 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.

6. 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 **I2SRXCFG** register.
 - Transmitter: Masters the **I2S0TXSCK** by ORing 0x0040.0000 into the **I2STXCFG** register.
7. 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 **I2STXCFG** and **I2SRXCFG** registers. The settings are shown in the table below.

Table 16-9. Audio Formats Configuration

Audio Format	I2STXCFG/I2SRXCFG Register Bit			
	JST	DLY	SCP	LRP
I ² 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.
- Set up the FIFO limits for triggering interrupts (also used for μDMA)
 - 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.
 - 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.
 - Enable the Serial Encoder and Serial Decoders by writing a 0x0000.0003 to the **I2SCFG** register.

16.5 Register Map

Table 16-10 on page 779 lists the I²S registers. The offset listed is a hexadecimal increment to the register's address, relative to the I²S interface base address of 0x4005.4000. Note that the I²S module clock must be enabled before the registers can be programmed (see page 276).

Table 16-10. Inter-Integrated Circuit Sound (I²S) Interface Register Map

Offset	Name	Type	Reset	Description	See page
0x000	I2STXFIFO	WO	0x0000.0000	I2S Transmit FIFO Data	780
0x004	I2STXFIFOCFG	R/W	0x0000.0000	I2S Transmit FIFO Configuration	781
0x008	I2STXCFG	R/W	0x1400.7DF0	I2S Transmit Module Configuration	782
0x00C	I2STXLIMIT	R/W	0x0000.0000	I2S Transmit FIFO Limit	784
0x010	I2STXISM	R/W	0x0000.0000	I2S Transmit Interrupt Status and Mask	785
0x018	I2STXLEV	RO	0x0000.0000	I2S Transmit FIFO Level	786
0x800	I2SRXFIFO	RO	0x0000.0000	I2S Receive FIFO Data	787
0x804	I2SRXFIFOCFG	R/W	0x0000.0000	I2S Receive FIFO Configuration	788
0x808	I2SRXCFG	R/W	0x1400.7DF0	I2S Receive Module Configuration	789
0x80C	I2SRXLIMIT	R/W	0x0000.7FFF	I2S Receive FIFO Limit	791
0x810	I2SRXISM	R/W	0x0000.0000	I2S Receive Interrupt Status and Mask	792
0x818	I2SRXLEV	RO	0x0000.0000	I2S Receive FIFO Level	793
0xC00	I2SCFG	R/W	0x0000.0000	I2S Module Configuration	794
0xC10	I2SIM	R/W	0x0000.0000	I2S Interrupt Mask	796
0xC14	I2SRIS	RO	0x0000.0000	I2S Raw Interrupt Status	798
0xC18	I2SMIS	RO	0x0000.0000	I2S Masked Interrupt Status	800
0xC1C	I2SIC	WO	0x0000.0000	I2S Interrupt Clear	802

16.6 Register Descriptions

The remainder of this section lists and describes the I²S registers, in numerical order by address offset.

Register 1: I²S 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 **I²S Transmit FIFO Configuration (I2STXFIFO CFG)** 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.

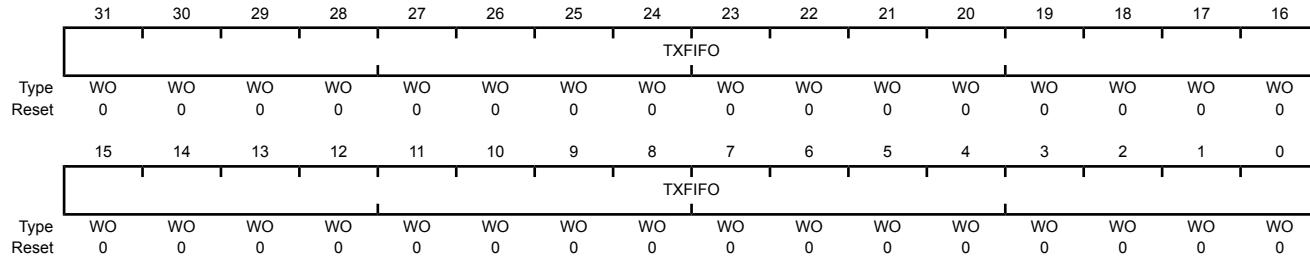
Note that if the FIFO is full and a write is attempted, a transmit FIFO write error is generated.

I2S Transmit FIFO Data (I2STXFIFO)

Base 0x4005.4000

Offset 0x000

Type WO, reset 0x0000.0000



Bit/Field Name Type Reset Description

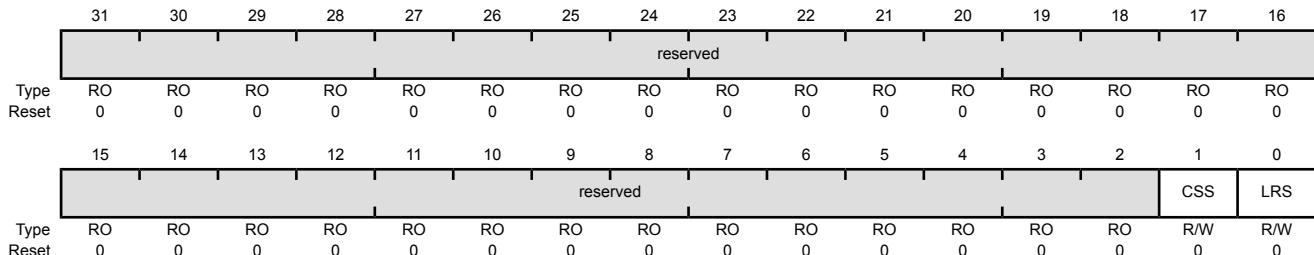
31:0 TXFIFO WO 0x0000.0000 TX Data
Serial audio sample data to be transmitted.

Register 2: I²S Transmit FIFO Configuration (I2STXFIFO CFG), 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.

I2S Transmit FIFO Configuration (I2STXFIFO CFG)

Base 0x4005.4000
Offset 0x004
Type R/W, reset 0x0000.0000



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
		Value	Description	
		0	The transmitter is in Compact 16-bit Stereo Mode with a 16-bit sample size.	
		1	The transmitter is in Compact 8-bit Stereo Mode with an 8-bit sample size.	
0	LRS	R/W	0	Left-Right Sample Indicator
		Value	Description	
		0	The left sample is the next position.	
		1	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: I²S 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	RO	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	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 Output Data
		Value	Description	
		0	The data is Left-Justified.	
		1	The data is Right-Justified.	
28	DLY	R/W	1	Data Delay
		Value	Description	
		0	Data is latched on the next latching edge of I ₂ S0TXSCK as defined by the SCP bit. This bit should be clear in Left-Justified or Right-Justified mode.	
		1	A one-I ₂ S0TXSCK delay from the edge of I ₂ S0TXWS is inserted before data is latched. This bit should be set in I ² S mode.	
27	SCP	R/W	0	SCLK Polarity
		Value	Description	
		0	Data and the I ₂ S0TXWS signal (when the MSL bit is set) are launched on the falling edge of I ₂ S0TXSCK.	
		1	Data and the I ₂ S0TXWS signal (when the MSL bit is set) are launched on the rising edge of I ₂ S0TXSCK.	
26	LRP	R/W	1	Left/Right Clock Polarity
		Value	Description	
		0	I ₂ S0TXWS is high during the transmission of the left channel data.	
		1	I ₂ S0TXWS is high during the transmission of the right channel data.	

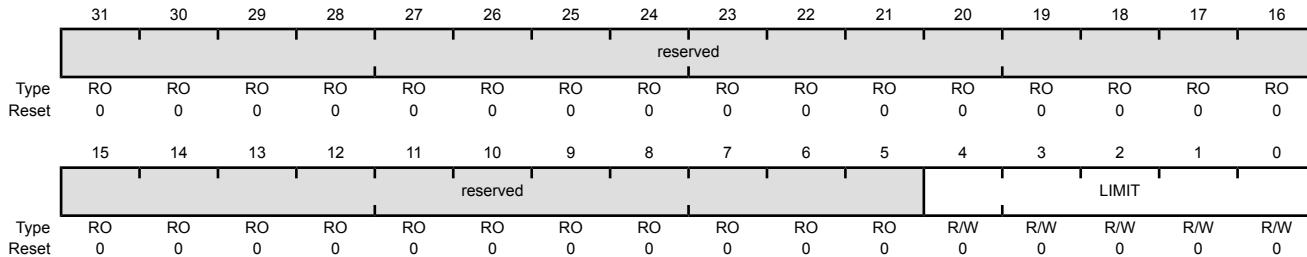
Bit/Field	Name	Type	Reset	Description										
25:24	WM	R/W	0x0	<p>Write Mode This bit field selects the mode in which the transmit data is stored in the FIFO and transmitted.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Stereo mode</td></tr> <tr> <td>0x1</td><td>Compact Stereo mode Left/Right sample packed. Refer to I2STXFIFO CFG 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 I2STXFIFO CFG 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 I2STXFIFO CFG for 8/16-bit sample size selection.													
0x2	Mono mode													
0x3	reserved													
23	FMT	R/W	0	<p>FIFO Empty</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>All zeroes are transmitted if the FIFO is empty.</td></tr> <tr> <td>1</td><td>The last sample is transmitted if the FIFO is empty.</td></tr> </tbody> </table>	Value	Description	0	All zeroes are transmitted if the FIFO is empty.	1	The last sample is transmitted if the FIFO is empty.				
Value	Description													
0	All zeroes are transmitted if the FIFO is empty.													
1	The last sample is transmitted if the FIFO is empty.													
22	MSL	R/W	0	<p>SCLK Master/Slave Source of serial bit clock (I2S0TXSCK) and Word Select (I2S0TXWS).</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The transmitter is a slave using the externally driven I2S0TXSCK and I2S0TXWS signals.</td></tr> <tr> <td>1</td><td>The transmitter is a master using the internally generated I2S0TXSCK and I2S0TXWS signals.</td></tr> </tbody> </table>	Value	Description	0	The transmitter is a slave using the externally driven I2S0TXSCK and I2S0TXWS signals.	1	The transmitter is a master using the internally generated I2S0TXSCK and I2S0TXWS signals.				
Value	Description													
0	The transmitter is a slave using the externally driven I2S0TXSCK and I2S0TXWS signals.													
1	The transmitter is a master using the internally generated I2S0TXSCK and I2S0TXWS signals.													
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 This field contains the number of bits minus one in the sample.</p> <p>Note: This field is only used in Right-Justified mode. Unused bits are not masked.</p>										
9:4	SDSZ	R/W	0x1F	<p>System Data Size This field contains the number of bits minus one during the high or low phase of the I2S0TXWS 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: I²S 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



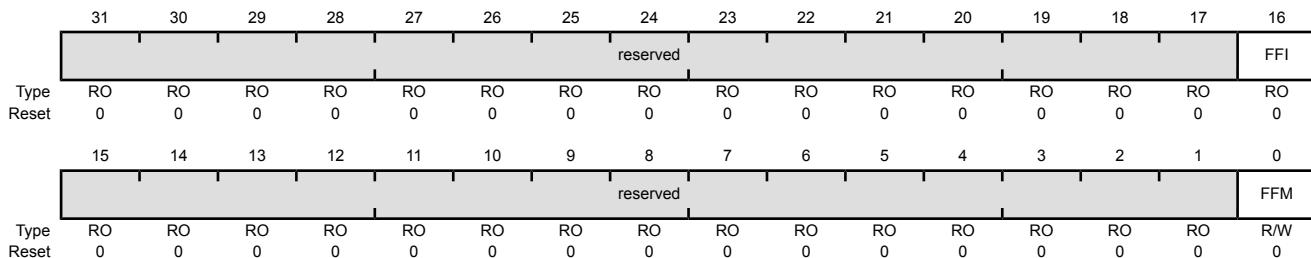
Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:0	LIMIT	R/W	0x00	FIFO Limit This field sets the FIFO level at which a FIFO service request is issued, generating an interrupt or a µDMA transfer request. The transmit FIFO generates a service request when the number of items in the FIFO is less than the level specified by the LIMIT field. For example, if the LIMIT field is set to 8, then a service request is generated when there are less than 8 samples remaining in the transmit FIFO.

Register 5: I²S 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



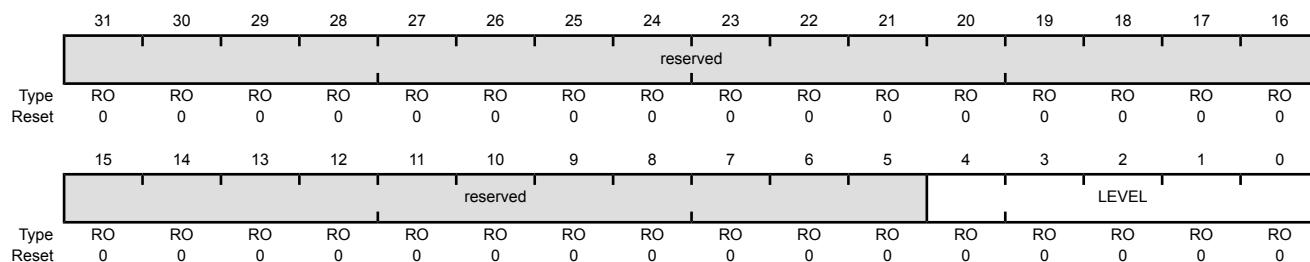
Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	FFI	RO	0	Transmit FIFO Service Request Interrupt
		Value	Description	
		0	The FIFO level is equal to or above the FIFO limit.	
		1	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
		Value	Description	
		0	The FIFO interrupt is masked and not sent to the CPU.	
		1	The FIFO interrupt is enabled to be sent to the interrupt controller.	

Register 6: I²S 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: I²S Receive FIFO Data (I2SRXFIFO), offset 0x800

Important: Use caution when reading this register. Performing a read may change bit status.

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 **I²S Receive FIFO Configuration (I2SRXFIFO CFG)** 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															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RXFIFO																
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	RXFIFO	RO	0x0000.0000	RX Data Serial audio sample data received. The read of an empty FIFO returns a value of 0x0.

Register 8: I²S Receive FIFO Configuration (I2SRXFIFO CFG), 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 (I2SRXFIFO CFG)

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
Type	RO	RO	RO													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO	R/W	R/W	R/W												
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	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	FMM	R/W	0	FIFO Mono Mode
		Value	Description	
		0	The receiver is in Stereo Mode.	
		1	The receiver is in Mono mode. If the LRP bit in the I2SRXCFG register is clear, data is read while the I2S0RXWS signal is low (Right Channel); if the LRP bit is set, data is read while the I2S0RXWS signal is high (Left Channel).	
1	CSS	R/W	0	Compact Stereo Sample Size
		Value	Description	
		0	The receiver is in Compact 16-bit Stereo Mode with a 16-bit sample size.	
		1	The receiver is in Compact 8-bit Stereo Mode with a 8-bit sample size.	
0	LRS	R/W	0	Left-Right Sample Indicator
		Value	Description	
		0	The left sample is the next position to be read.	
		1	The right sample is the next position to be read.	
		This bit is only meaningful in Compact Stereo Mode.		

Register 9: I²S 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	RO	R/W	RO	R/W	RO	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	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
		Value	Description	
		0	The data is Left-Justified.	
		1	The data is Right-Justified.	
28	DLY	R/W	1	Data Delay
		Value	Description	
		0	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.	
		1	A one-I2S0RXSCK delay from the edge of I2S0RXWS is inserted before data is latched. This bit should be set in I ² S mode.	
27	SCP	R/W	0	SCLK Polarity
		Value	Description	
		0	Data is latched on the rising edge and the I2S0RXWS signal (when the MSL bit is set) is launched on the falling edge of I2S0RXSCK.	
		1	Data is latched on the falling edge and the I2S0RXWS signal (when the MSL bit is set) is launched on the rising edge of I2S0RXSCK.	

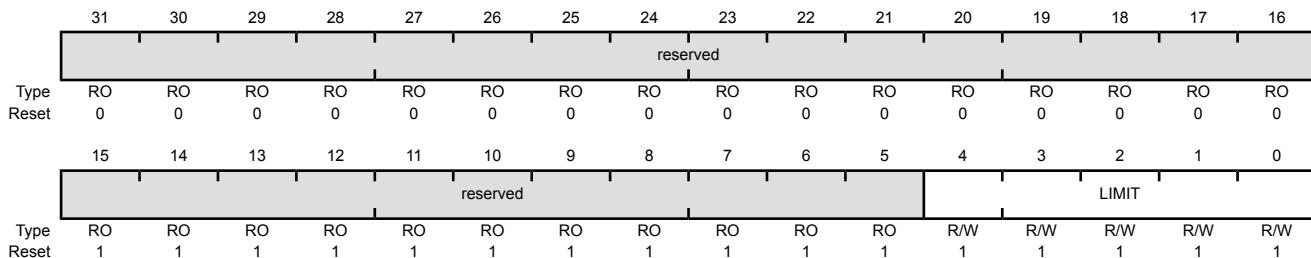
Bit/Field	Name	Type	Reset	Description
26	LRP	R/W	1	Left/Right Clock Polarity Value Description 0 In Stereo mode, I ₂ S0RXWS is high during the transmission of the left channel data. In Mono mode, data is read while the I ₂ S0RXWS signal is low (Right Channel). 1 In Stereo mode, I ₂ S0RXWS is high during the transmission of the right channel data. In Mono mode, data is read while the I ₂ S0RXWS signal is high (Left Channel).
25	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	RM	R/W	0	Read Mode This bit selects the mode in which the receive data is received and stored in the FIFO. Value Description 0 Stereo/Mono mode I ₂ SRXFIFO CFG FMM bit specifies Stereo or Mono FIFO read behavior. 1 Compact Stereo mode Left/Right sample packed. Refer to I ₂ SRXFIFO CFG for 8/16-bit sample size selection.
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	SCLK Master/Slave Value Description 0 The receiver is a slave and uses the externally driven I ₂ S0RXSCK and I ₂ S0RXWS signals. 1 The receiver is a master and uses the internally generated I ₂ S0RXSCK and I ₂ S0RXWS signals.
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	Sample Size This field contains the number of bits minus one in the sample.
9:4	SDSZ	R/W	0x1F	System Data Size This field contains the number of bits minus one during the high or low phase of the I ₂ S0RXWS signal.
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: I²S 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



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 μDMA 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 LIMIT field. For example, if the LIMIT field is set to 4, then a service request is generated when there are more than 4 samples remaining in the transmit FIFO.

Register 11: I²S 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	

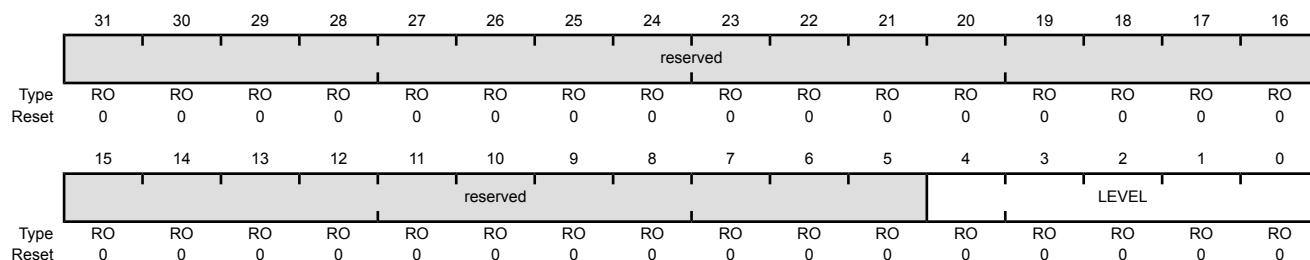
Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	FFI	RO	0	Receive FIFO Service Request Interrupt
		Value	Description	
		0	The FIFO level is equal to or below the FIFO limit.	
		1	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
		Value	Description	
		0	The FIFO interrupt is masked and not sent to the CPU.	
		1	The FIFO interrupt is enabled to be sent to the interrupt controller.	

Register 12: I²S 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



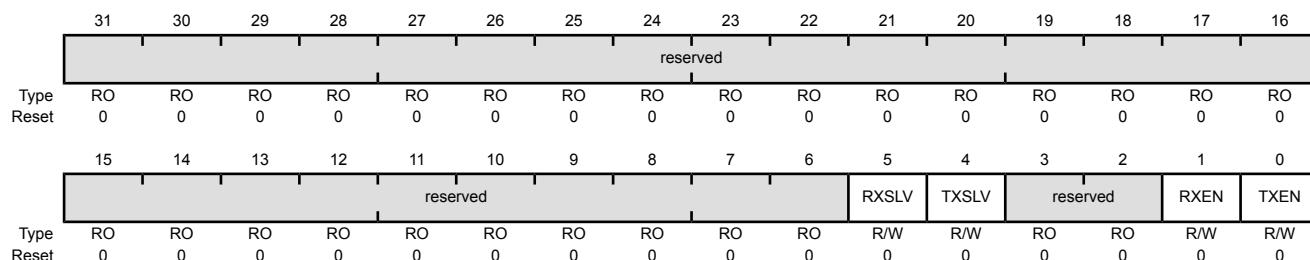
Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:0	LEVEL	RO	0x00	Number of Audio Samples This field contains the number of samples in the FIFO.

Register 13: I²S 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



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 I ₂ S0RXMCLK
		Value	Description	
		0	The receiver uses the internally generated MCLK as the I ₂ S0RXMCLK signal. See "Clock Control" on page 772 for information on how to program the I ₂ S0RXMCLK.	
		1	The receiver uses the externally driven I ₂ S0RXMCLK signal.	
4	TXSLV	R/W	0	Use External I ₂ S0TXMCLK
		Value	Description	
		0	The transmitter uses the internally generated MCLK as the I ₂ S0TXMCLK signal. See "Clock Control" on page 772 for information on how to program the I ₂ S0TXMCLK.	
		1	The transmitter uses the externally driven I ₂ S0TXMCLK signal.	
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
		Value	Description	
		0	Disables the serial receive engine.	
		1	Enables the serial receive engine.	

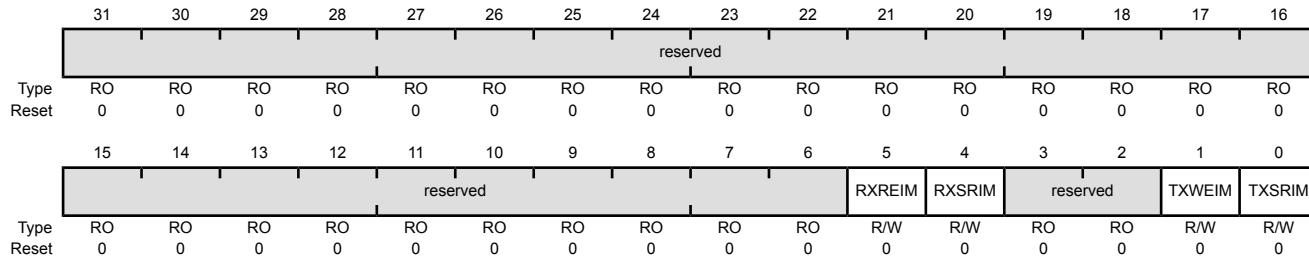
Bit/Field	Name	Type	Reset	Description
0	TXEN	R/W	0	Serial Transmit Engine Enable
Value Description				
		0		Disables the serial transmit engine.
		1		Enables the serial transmit engine.

Register 14: I²S 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



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	RXREIM	R/W	0	Receive FIFO Read Error
		Value	Description	
		0	The receive FIFO read error interrupt is masked and not sent to the CPU.	
		1	The receive FIFO read error is enabled to be sent to the interrupt controller.	
4	RXSRIM	R/W	0	Receive FIFO Service Request
		Value	Description	
		0	The receive FIFO service request interrupt is masked and not sent to the CPU.	
		1	The receive FIFO service request is enabled to be sent to the interrupt controller.	
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	TXWEIM	R/W	0	Transmit FIFO Write Error
		Value	Description	
		0	The transmit FIFO write error interrupt is masked and not sent to the CPU.	
		1	The transmit FIFO write error is enabled to be sent to the interrupt controller.	

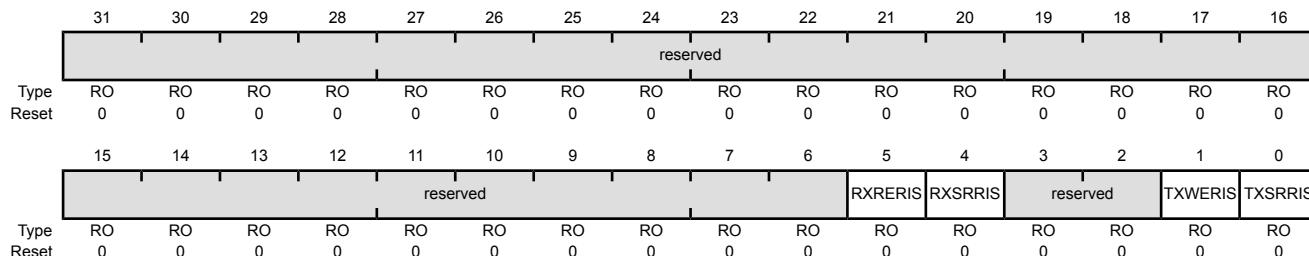
Bit/Field	Name	Type	Reset	Description
0	TXSRIM	R/W	0	Transmit FIFO Service Request
Value Description				
		0		The transmit FIFO service request interrupt is masked and not sent to the CPU.
		1		The transmit FIFO service request is enabled to be sent to the interrupt controller.

Register 15: I²S 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



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	RXRERIS	RO	0	Receive FIFO Read Error
		Value	Description	
		1	A receive FIFO read error interrupt has occurred.	
		0	No interrupt	
		This bit is cleared by setting the RXREIC bit in the I2SIC register.		
4	RXSRRIS	RO	0	Receive FIFO Service Request
		Value	Description	
		1	A receive FIFO service request interrupt has occurred.	
		0	No interrupt	
		This bit is cleared when the level in the receive FIFO has risen to a value greater than the value programmed in the LIMIT field in the I2SRXLIMIT 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	TXWERIS	RO	0	Transmit FIFO Write Error
		Value	Description	
		1	A transmit FIFO write error interrupt has occurred.	
		0	No interrupt	
		This bit is cleared by setting the TXWEIC bit in the I2SIC register.		

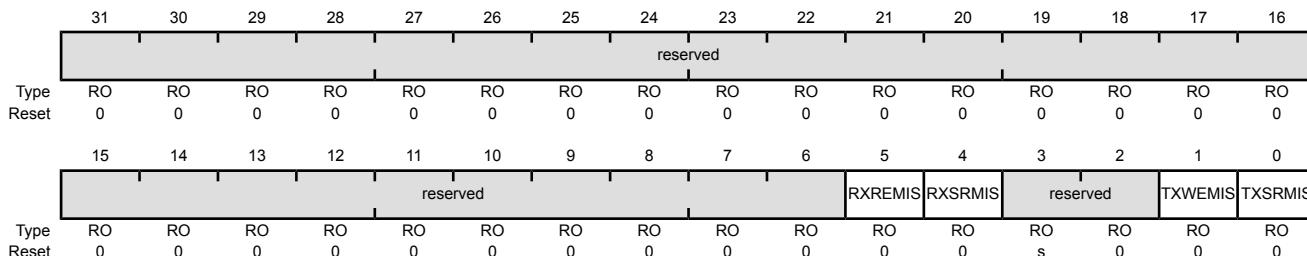
Bit/Field	Name	Type	Reset	Description
0	TXSRRIS	RO	0	Transmit FIFO Service Request
Value Description				
1 A transmit FIFO service request interrupt has occurred.				
0 No interrupt				
This bit is cleared when the level in the transmit FIFO has fallen to a value less than the value programmed in the LIMIT field in the I2STXLIMIT register.				

Register 16: I²S 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



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	RXREMIS	RO	0	Receive FIFO Read Error
		Value	Description	
		1	An unmasked interrupt was signaled due to a receive FIFO read error.	
		0	An interrupt has not occurred or is masked.	
		This bit is cleared by setting the RXREIC bit in the I2SIC register.		
4	RXSRMIS	RO	0	Receive FIFO Service Request
		Value	Description	
		1	An unmasked interrupt was signaled due to a receive FIFO service request.	
		0	An interrupt has not occurred or is masked.	
		This bit is cleared when the level in the receive FIFO has risen to a value greater than the value programmed in the LIMIT field in the I2SRXLIMIT register.		
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	TXWEMIS	RO	0	Transmit FIFO Write Error
		Value	Description	
		1	An unmasked interrupt was signaled due to a transmit FIFO write error.	
		0	An interrupt has not occurred or is masked.	
		This bit is cleared by setting the TXWEIC bit in the I2SIC register.		

Bit/Field	Name	Type	Reset	Description
0	TXSRMIS	RO	0	Transmit FIFO Service Request Value Description 1 An unmasked interrupt was signaled due to a transmit FIFO service request. 0 An interrupt has not occurred or is masked. This bit is cleared when the level in the transmit FIFO has fallen to a value less than the value programmed in the LIMIT field in the I2STXLIMIT register.

Register 17: I²S Interrupt Clear (I2SIC), offset 0xC1C

Writing a 1 to 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										RXREIC	reserved		TXWEIC	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	RXREIC	WO	0	Receive FIFO Read Error Writing a 1 to this bit clears the RXRERIS bit in the I2CRIS register and the RXREMIS bit in the I2CMIS 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	TXWEIC	WO	0	Transmit FIFO Write Error Writing a 1 to this bit clears the TXWERIS bit in the I2CRIS register and the TXWEMIS bit in the I2CMIS 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.

17 Controller Area Network (CAN) Module

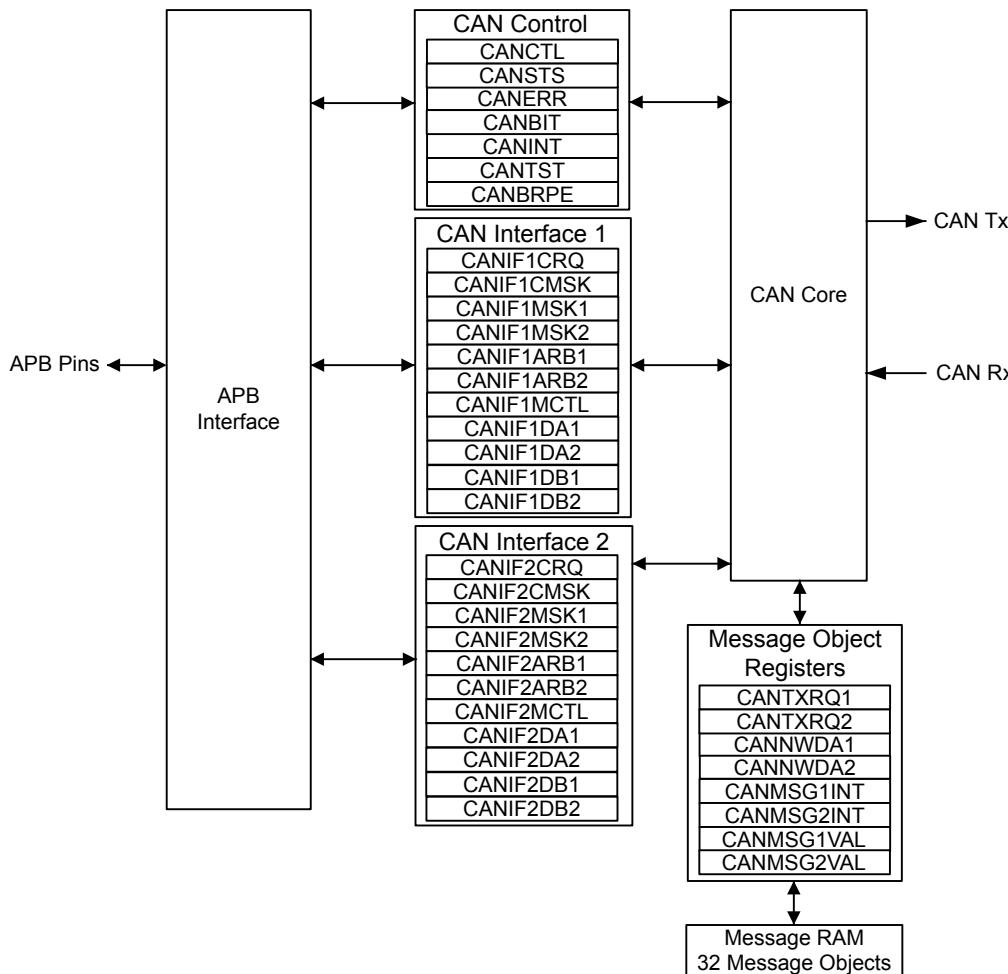
Controller Area Network (CAN) is a multicast, shared serial bus standard for connecting electronic control units (ECUs). CAN was specifically designed to be robust in electromagnetically-noisy environments and can utilize a differential balanced line like RS-485 or a more robust twisted-pair wire. Originally 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).

The Stellaris® LM3S9L97 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

17.1 Block Diagram

Figure 17-1. CAN Controller Block Diagram



17.2 Signal Description

Table 17-1 on page 805 and Table 17-2 on page 805 list the external signals of the CAN controller and describe the function of each. The CAN controller signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the CAN signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 447) should be set to choose the CAN controller function. The number in parentheses is the encoding that must be programmed into the PMC_n field in the **GPIO Port Control (GPIOPCTL)** register (page 465) to assign the CAN signal to the specified GPIO port pin. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 423.

Table 17-1. Signals for Controller Area Network (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
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.

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

Table 17-2. Signals for Controller Area Network (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
CAN0Rx	G1 L5 L6 A6	PD0 (2) PA4 (5) PA6 (6) PB4 (5)	I	TTL	CAN module 0 receive.
CAN0Tx	G2 M5 M6 B7	PD1 (2) PA5 (5) PA7 (6) PB5 (5)	O	TTL	CAN module 0 transmit.
CAN1Rx	M9	PF0 (1)	I	TTL	CAN module 1 receive.
CAN1Tx	H12	PF1 (1)	O	TTL	CAN module 1 transmit.

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

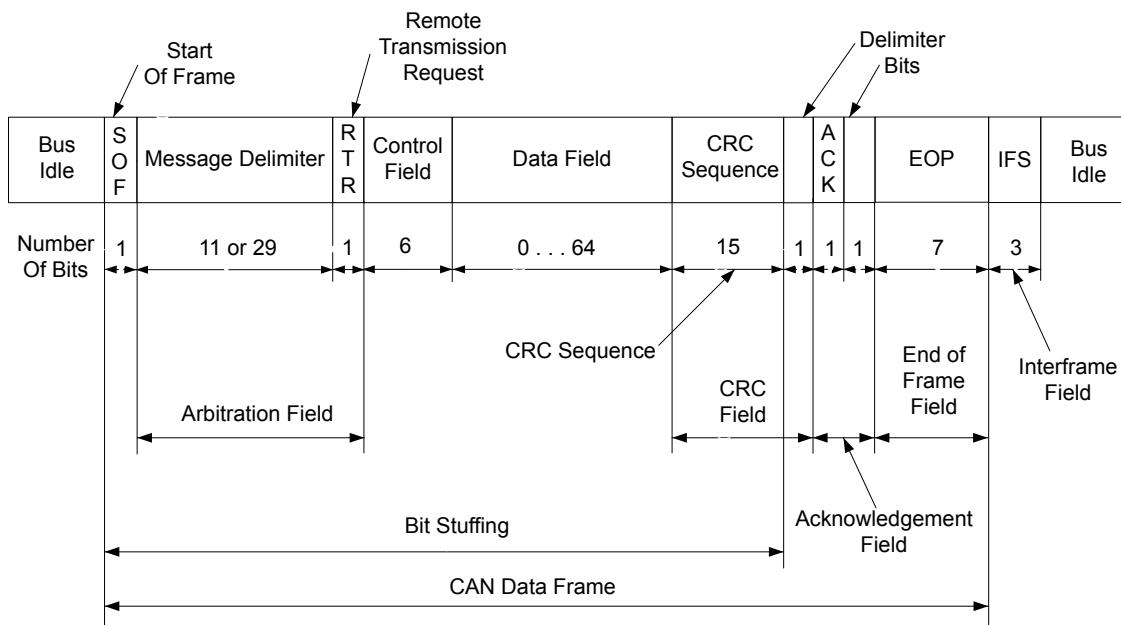
17.3 Functional Description

The Stellaris 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 17-2.

Figure 17-2. 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 memory blocks 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. The message object memory cannot be directly accessed, so 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.

17.3.1 Initialization

To use the CAN controller, the peripheral clock must be enabled using the **RCGC0** register (see page 268). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register (see page 285). To find out which GPIO port to enable, refer to Table 24-4 on page 1182. Set the GPIO AFSEL bits for the appropriate pins (see page 447). Configure the PMC_n fields in the **GPIOPCTL** register to assign the CAN signals to the appropriate pins. See page 465 and Table 24-5 on page 1190.

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.

17.3.2 Operation

Two sets of CAN Interface Registers (**CANIFn1x** and **CANIFn2x**) 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. Message objects can be used for one-time data transfers or can be 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. A remote frame 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.

17.3.3 Transmitting Message Objects

If the internal transmit shift register of the CAN module is ready for loading, and if a data transfer is not 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 configured to interrupt on 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.

17.3.4 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 on the reception of a matching remote frame allowing automatic transmission
 - Set the **EOB** bit for a single message object
 - Configure 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)** 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.

17.3.5 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.

17.3.6 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.

17.3.7 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 on 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.

17.3.8 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:

Table 17-3. Message Object Configurations

Configuration in CANIFnMCTL	Description
<ul style="list-style-type: none"> ■ DIR = 1 (direction = transmit); programmed in the CANIFnARB2 register ■ RMTEN = 1 (set the TXRQST bit of the CANIFnMCTL register at reception of the frame to enable transmission) ■ UMASK = 1 or 0 	At the reception of a matching remote frame, the TXRQST bit of this message object is set. The rest of the message object remains unchanged, and the controller automatically transfers the data in the message object as soon as possible.
<ul style="list-style-type: none"> ■ DIR = 1 (direction = transmit); programmed in the CANIFnARB2 register ■ RMTEN = 0 (do not change the TXRQST bit of the CANIFnMCTL register at reception of the frame) ■ UMASK = 0 (ignore mask in the CANIFnMSKn register) 	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 nothing indicates that the remote frame ever happened.
<ul style="list-style-type: none"> ■ DIR = 1 (direction = transmit); programmed in the CANIFnARB2 register ■ RMTEN = 0 (do not change the TXRQST bit of the CANIFnMCTL register at reception of the frame) ■ UMASK = 1 (use mask (MSK, MXTD, and MDIR in the CANIFnMSKn register) for acceptance filtering) 	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 mode 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.

17.3.9 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 prioritization is separate from that of the message identifier which is enforced by the CAN bus. As a result, if message object 1 and message object 2 both have valid messages to be transmitted, message object 1 is always transmitted first regardless of the message identifier in the message object itself.

17.3.10 Configuring a Receive Message Object

The following steps illustrate how to configure a receive message object.

1. Program the **CAN IFn Command Mask (CANIFnCMASK)** register as described in the “Configuring a Transmit Message Object” on page 808 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 808 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 808 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
 - Configure 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.

17.3.11 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.

17.3.11.1 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 811). 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.

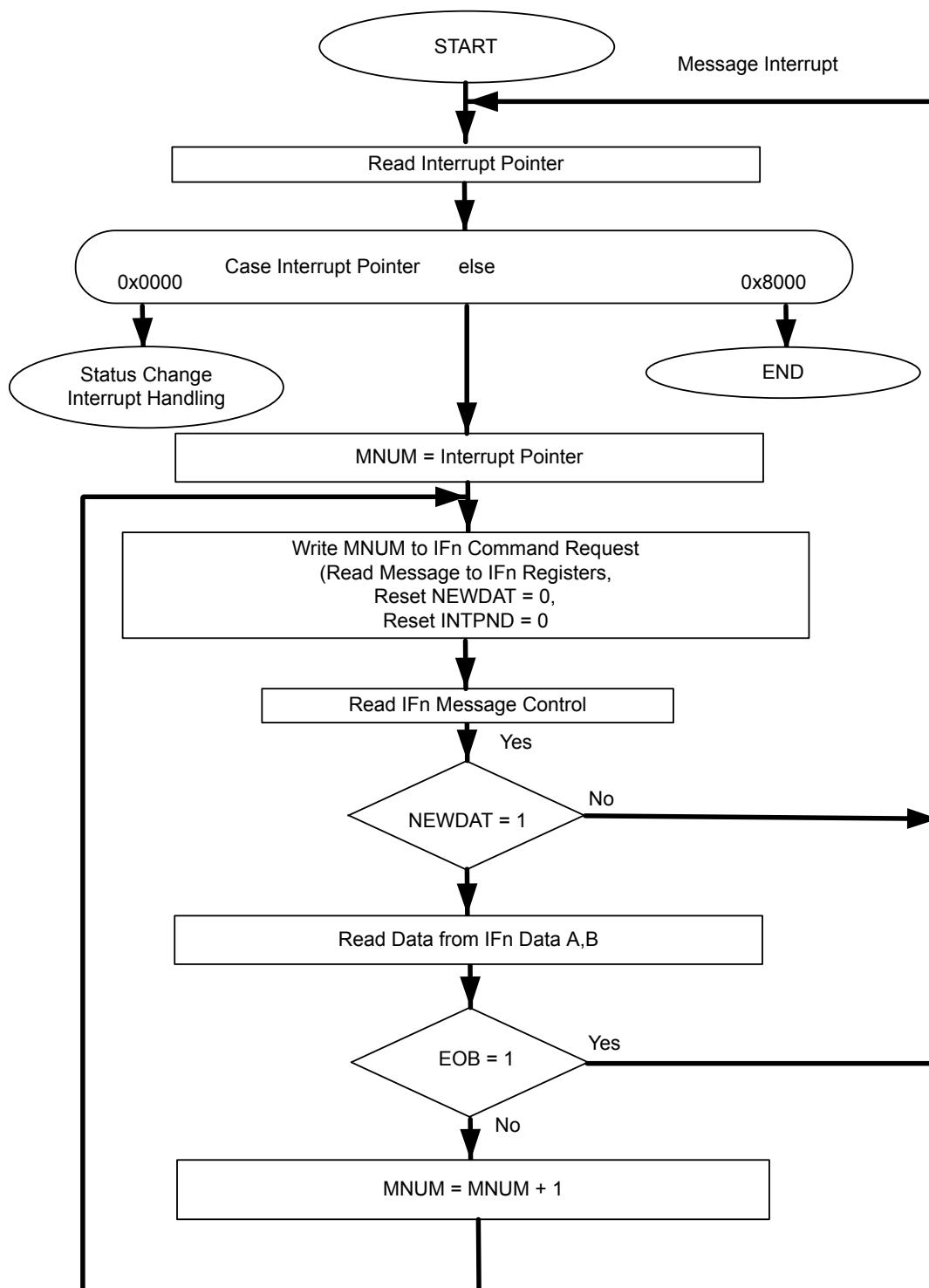
17.3.11.2 Reception of Messages with FIFO Buffers

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. Until all of the preceding message objects have been released by clearing the NEWDAT bit, all further messages for this FIFO buffer are written into the last message object of the FIFO buffer and therefore overwrite previous messages.

17.3.11.3 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. When reading from the FIFO buffer, the user should be aware that a new received message could be placed in the location of any message object for which the NEWDAT bit of the **CANIFnMCTL** register is clear. As a result, the order of the received messages in the FIFO is not guaranteed. Figure 17-3 on page 814 shows how a set of message objects which are concatenated to a FIFO Buffer can be handled by the CPU.

Figure 17-3. Message Objects in a FIFO Buffer



17.3.12 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 an interrupt is pending. If the IE bit is set in the **CANCTL** register, the interrupt line to the interrupt controller 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 interrupt controller. The **CANINT** register is updated even when the IE bit in the **CANCTL** register is clear, but the interrupt is not 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.

The source of an interrupt can be determined in two ways 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.

17.3.13 Test Mode

A Test Mode is provided which allows various diagnostics to be performed. Test Mode is entered by setting the TEST bit in the **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.

17.3.13.1 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 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.

17.3.13.2 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.

17.3.13.3 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.

17.3.13.4 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 are stored in 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 MSGLST 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.

17.3.13.5 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.

17.3.14 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.

17.3.15 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 17-4 on page 818): 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 17-4 on page 818). 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 input clock (f_{sys}) and the Baud Rate Prescaler (BRP):

$$t_q = BRP / f_{sys}$$

The f_{sys} input clock is the system clock frequency as configured by the **RCC** or **RCC2** registers (see page 226 or page 234).

The Synchronization Segment Sync 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 and the Sync is called the phase error of that edge.

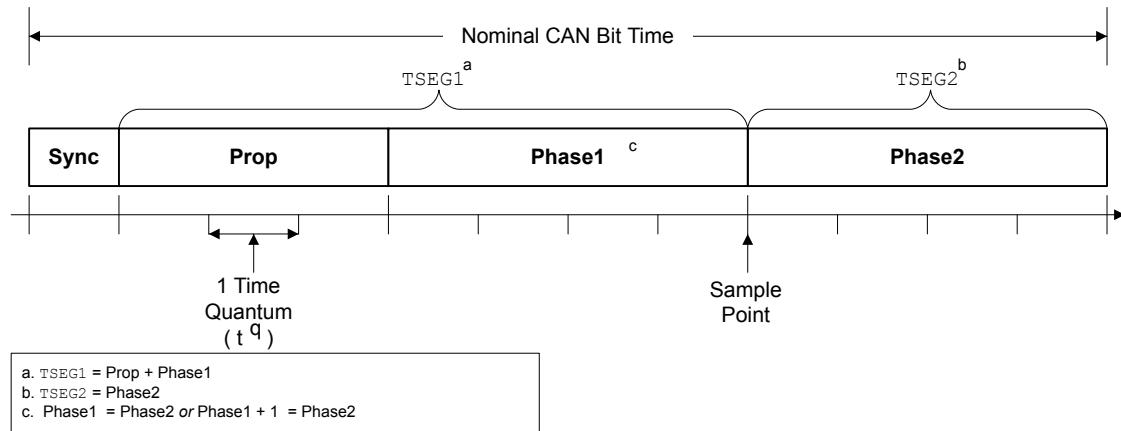
The Propagation Time Segment Prop is intended to compensate for the physical delay times within the CAN network.

The Phase Buffer Segments Phase1 and Phase2 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 17-4. CAN Bit Time

Table 17-4. CAN Protocol Ranges^a

Parameter	Range	Remark
BRP	[1 .. 64]	Defines the length of the time quantum t_q . The CANBRPE register can be used to extend the range to 1024.
Sync	$1 t_q$	Fixed length, synchronization of bus input to system clock
Prop	[1 .. 8] t_q	Compensates for the physical delay times
Phase1	[1 .. 8] t_q	May be lengthened temporarily by synchronization
Phase2	[1 .. 8] t_q	May be shortened temporarily by synchronization
SJW	[1 .. 4] t_q	May not be longer than either Phase Buffer Segment

a. This table describes the minimum programmable ranges required by the CAN protocol.

The bit timing configuration is programmed in two register bytes in the **CANBIT** register. In the **CANBIT** register, the four components TSEG2, TSEG1, 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 in the SJW bit field. Table 17-5 shows the relationship between the **CANBIT** register values and the parameters.

Table 17-5. CANBIT Register Values

CANBIT Register Field	Setting
TSEG2	Phase2 - 1
TSEG1	Prop + Phase1 - 1
SJW	SJW - 1
BRP	BRP

Therefore, the length of the bit time is (programmed values):

$$[TSEG1 + TSEG2 + 3] \times t_q$$

or (functional values):

$$[Sync + Prop + Phase1 + Phase2] \times t_q$$

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. In addition, the controller 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. The bit value is received or transmitted at the sample point. The information processing time (IPT) is the time after the sample point needed to calculate the next bit to be transmitted on the CAN bus. The IPT includes any of the following: retrieving the next data bit, handling a CRC bit, determining if bit stuffing is required, generating an error flag or simply going idle.

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 Phase2. In case of synchronization, Phase2 may be shortened to a value less than IPT, which does not affect bus timing.

17.3.16 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 Prop. 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 is converted into time quanta (rounded up to the nearest integer multiple of t_q).

Sync is $1 t_q$ long (fixed), which leaves (bit time - Prop - 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, Phase2 = Phase1, else Phase2 = Phase1 + 1.

The minimum nominal length of Phase2 has to be regarded as well. Phase2 may not be shorter than the CAN controller's Information Processing Time, which is, depending on the actual implementation, in the range of [0..2] t_q .

The length of the synchronization jump width is set to the least of 4, Phase1 or Phase2.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formula given below:

$$(1 - df) \times f_{nom} \leq f_{osc} \leq (1 + df) \times f_{nom}$$

where:

- df = Maximum tolerance of oscillator frequency
- f_{osc} = Actual oscillator frequency
- f_{nom} = Nominal oscillator frequency

Maximum frequency tolerance must take into account the following formulas:

$$df \leq \frac{(Phase_seg1, Phase_seg2) \min}{2 \times (13 \times tbit - Phase_Seg2)}$$

$$df \max = 2 \times df \times f_{nom}$$

where:

- Phase1 and Phase2 are from Table 17-4 on page 818
- 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.

17.3.16.1 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.

```

bit time = 1 μs = n * tq = 5 * tq
tq = 200 ns
tq = (Baud rate Prescaler)/CAN Clock
Baud rate Prescaler = tq * CAN Clock
Baud rate Prescaler = 200E-9 * 25E6 = 5

tSync = 1 * tq = 200 ns          \\fixed at 1 time quanta

delay of bus driver 50 ns
delay of receiver circuit 30 ns
delay of bus line (40m) 220 ns
tProp 400 ns = 2 * tq          \\400 is next integer multiple of tq

bit time = tSync + tTSeg1 + tTSeg2 = 5 * tq
bit time = tSync + tProp + tPhase 1 + tPhase2
tPhase 1 + tPhase2 = bit time - tSync - tProp
tPhase 1 + tPhase2 = (5 * tq) - (1 * tq) - (2 * tq)
tPhase 1 + tPhase2 = 2 * tq
tPhase1 = 1 * tq
tPhase2 = 1 * tq              \\tPhase2 = tPhase1

```

```

tTSeg1 = tProp + tPhasel
tTSeg1 = (2 * tq) + (1 * tq)
tTSeg1 = 3 * tq

tTSeg2 = tPhase2
tTSeg2 = (Information Processing Time + 1) * tq
tTSeg2 = 1 * tq                                \\Assumes IPT=0

tSJW = 1 * tq                                \\Least of 4, Phasel and Phase2

```

In the above example, the bit field values for the **CANBIT** register are:

TSEG2	= TSeg2 -1 = 1-1 = 0
TSEG1	= TSeg1 -1 = 3-1 = 2
SJW	= SJW -1 = 1-1 = 0
BRP	= Baud rate prescaler - 1 = 5-1 =4

The final value programmed into the **CANBIT** register = 0x0204.

17.3.16.2 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.

```

bit time = 10 μs = n * tq = 10 * tq
tq = 1 μs
tq = (Baud rate Prescaler)/CAN Clock
Baud rate Prescaler = tq * CAN Clock
Baud rate Prescaler = 1E-6 * 50E6 = 50

tSync = 1 * tq = 1 μs                      \\fixed at 1 time quanta

delay of bus driver 200 ns
delay of receiver circuit 80 ns
delay of bus line (40m) 220 ns
tProp 1 μs = 1 * tq                         \\1 μs is next integer multiple of tq

bit time = tSync + tTSeg1 + tTSeg2 = 10 * tq
bit time = tSync + tProp + tPhase1 + tPhase2
tPhase1 + tPhase2 = bit time - tSync - tProp
tPhase1 + tPhase2 = (10 * tq) - (1 * tq) - (1 * tq)
tPhase1 + tPhase2 = 8 * tq
tPhase1 = 4 * tq
tPhase2 = 4 * tq                            \\tPhase1 = tPhase2

tTSeg1 = tProp + tPhasel

```

```

tTSeg1 = (1 * tq) + (4 * tq)
tTSeg1 = 5 * tq
tTSeg2 = tPhase2
tTSeg2 = (Information Processing Time + 4) × tq
tTSeg2 = 4 * tq                                \\Assumes IPT=0

tSJW = 4 * tq                                \\Least of 4, Phase1, and Phase2

```

TSEG2	= TSeg2 -1 = 4-1 = 3
TSEG1	= TSeg1 -1 = 5-1 = 4
SJW	= SJW -1 = 4-1 = 3
BRP	= Baud rate prescaler - 1 = 50-1 =49

The final value programmed into the **CANBIT** register = 0x34F1.

17.4 Register Map

Table 17-6 on page 822 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 268).

Table 17-6. CAN Register Map

Offset	Name	Type	Reset	Description	See page
0x000	CANCTL	R/W	0x0000.0001	CAN Control	824
0x004	CANSTS	R/W	0x0000.0000	CAN Status	826
0x008	CANERR	RO	0x0000.0000	CAN Error Counter	829
0x00C	CANBIT	R/W	0x0000.2301	CAN Bit Timing	830
0x010	CANINT	RO	0x0000.0000	CAN Interrupt	831
0x014	CANTST	R/W	0x0000.0000	CAN Test	832
0x018	CANBRPE	R/W	0x0000.0000	CAN Baud Rate Prescaler Extension	834
0x020	CANIF1CRQ	R/W	0x0000.0001	CAN IF1 Command Request	835
0x024	CANIF1CMSK	R/W	0x0000.0000	CAN IF1 Command Mask	836
0x028	CANIF1MSK1	R/W	0x0000.FFFF	CAN IF1 Mask 1	839

Table 17-6. CAN Register Map (*continued*)

Offset	Name	Type	Reset	Description	See page
0x02C	CANIF1MSK2	R/W	0x0000.FFFF	CAN IF1 Mask 2	840
0x030	CANIF1ARB1	R/W	0x0000.0000	CAN IF1 Arbitration 1	842
0x034	CANIF1ARB2	R/W	0x0000.0000	CAN IF1 Arbitration 2	843
0x038	CANIF1MCTL	R/W	0x0000.0000	CAN IF1 Message Control	845
0x03C	CANIF1DA1	R/W	0x0000.0000	CAN IF1 Data A1	848
0x040	CANIF1DA2	R/W	0x0000.0000	CAN IF1 Data A2	848
0x044	CANIF1DB1	R/W	0x0000.0000	CAN IF1 Data B1	848
0x048	CANIF1DB2	R/W	0x0000.0000	CAN IF1 Data B2	848
0x080	CANIF2CRQ	R/W	0x0000.0001	CAN IF2 Command Request	835
0x084	CANIF2CMSK	R/W	0x0000.0000	CAN IF2 Command Mask	836
0x088	CANIF2MSK1	R/W	0x0000.FFFF	CAN IF2 Mask 1	839
0x08C	CANIF2MSK2	R/W	0x0000.FFFF	CAN IF2 Mask 2	840
0x090	CANIF2ARB1	R/W	0x0000.0000	CAN IF2 Arbitration 1	842
0x094	CANIF2ARB2	R/W	0x0000.0000	CAN IF2 Arbitration 2	843
0x098	CANIF2MCTL	R/W	0x0000.0000	CAN IF2 Message Control	845
0x09C	CANIF2DA1	R/W	0x0000.0000	CAN IF2 Data A1	848
0x0A0	CANIF2DA2	R/W	0x0000.0000	CAN IF2 Data A2	848
0x0A4	CANIF2DB1	R/W	0x0000.0000	CAN IF2 Data B1	848
0x0A8	CANIF2DB2	R/W	0x0000.0000	CAN IF2 Data B2	848
0x100	CANTXRQ1	RO	0x0000.0000	CAN Transmission Request 1	849
0x104	CANTXRQ2	RO	0x0000.0000	CAN Transmission Request 2	849
0x120	CANNWDA1	RO	0x0000.0000	CAN New Data 1	850
0x124	CANNWDA2	RO	0x0000.0000	CAN New Data 2	850
0x140	CANMSG1INT	RO	0x0000.0000	CAN Message 1 Interrupt Pending	851
0x144	CANMSG2INT	RO	0x0000.0000	CAN Message 2 Interrupt Pending	851
0x160	CANMSG1VAL	RO	0x0000.0000	CAN Message 1 Valid	852
0x164	CANMSG2VAL	RO	0x0000.0000	CAN Message 2 Valid	852

17.5 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.

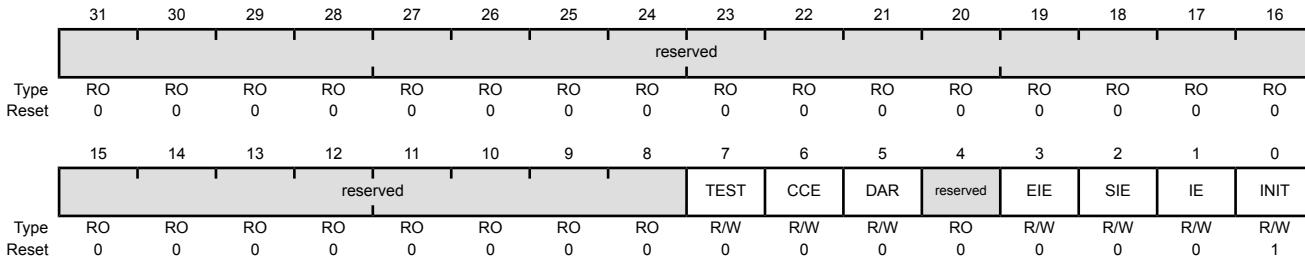
CAN Control (CANCTL)

CAN0 base: 0x4004.0000

CAN1 base: 0x4004.1000

Offset 0x000

Type R/W, reset 0x0000.0001



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 Value Description 0 The CAN controller is operating normally. 1 The CAN controller is in test mode.
6	CCE	R/W	0	Configuration Change Enable Value Description 0 Write accesses to the CANBIT register are not allowed. 1 Write accesses to the CANBIT register are allowed if the INIT bit is 1.
5	DAR	R/W	0	Disable Automatic-Retransmission Value Description 0 Auto-retransmission of disturbed messages is enabled. 1 Auto-retransmission is disabled.

Bit/Field	Name	Type	Reset	Description						
4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
3	EIE	R/W	0	Error Interrupt Enable						
				<table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No error status interrupt is generated.</td> </tr> <tr> <td>1</td> <td>A change in the BOFF or EWARN bits in the CANSTS register generates an interrupt.</td> </tr> </tbody> </table>	Value	Description	0	No error status interrupt is generated.	1	A change in the BOFF or EWARN bits in the CANSTS register generates an interrupt.
Value	Description									
0	No error status interrupt is generated.									
1	A change in the BOFF or EWARN bits in the CANSTS register generates an interrupt.									
2	SIE	R/W	0	Status Interrupt Enable						
				<table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>No status interrupt is generated.</td> </tr> <tr> <td>1</td> <td>An interrupt is generated when a message has successfully been transmitted or received, or a CAN bus error has been detected. A change in the TXOK, RXOK or LEC bits in the CANSTS register generates an interrupt.</td> </tr> </tbody> </table>	Value	Description	0	No status interrupt is generated.	1	An interrupt is generated when a message has successfully been transmitted or received, or a CAN bus error has been detected. A change in the TXOK, RXOK or LEC bits in the CANSTS register generates an interrupt.
Value	Description									
0	No status interrupt is generated.									
1	An interrupt is generated when a message has successfully been transmitted or received, or a CAN bus error has been detected. A change in the TXOK, RXOK or LEC bits in the CANSTS register generates an interrupt.									
1	IE	R/W	0	CAN Interrupt Enable						
				<table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Interrupts disabled.</td> </tr> <tr> <td>1</td> <td>Interrupts enabled.</td> </tr> </tbody> </table>	Value	Description	0	Interrupts disabled.	1	Interrupts enabled.
Value	Description									
0	Interrupts disabled.									
1	Interrupts enabled.									
0	INIT	R/W	1	Initialization						
				<table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Normal operation.</td> </tr> <tr> <td>1</td> <td>Initialization started.</td> </tr> </tbody> </table>	Value	Description	0	Normal operation.	1	Initialization started.
Value	Description									
0	Normal operation.									
1	Initialization started.									

Register 2: CAN Status (CANSTS), offset 0x004

Important: Use caution when reading this register. Performing a read may change bit status.

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 0x7 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								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	BOFF	EWARN	EPASS	RXOK	TXOK	LEC									
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
				Value Description
			0	The CAN controller is not in bus-off state.
			1	The CAN controller is in bus-off state.
6	EWARN	RO	0	Warning Status
				Value Description
			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.

Bit/Field	Name	Type	Reset	Description						
5	EPASS	RO	0	<p>Error Passive</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The CAN module is in the Error Active state, that is, the receive or transmit error count is less than or equal to 127.</td></tr> <tr> <td>1</td><td>The CAN module is in the Error Passive state, that is, the receive or transmit error count is greater than 127.</td></tr> </tbody> </table>	Value	Description	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.
Value	Description									
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.									
4	RXOK	R/W	0	<p>Received a Message Successfully</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Since this bit was last cleared, no message has been successfully received.</td></tr> <tr> <td>1</td><td>Since this bit was last cleared, a message has been successfully received, independent of the result of the acceptance filtering.</td></tr> </tbody> </table> <p>This bit must be cleared by writing a 0 to it.</p>	Value	Description	0	Since this bit was last cleared, no message has been successfully received.	1	Since this bit was last cleared, a message has been successfully received, independent of the result of the acceptance filtering.
Value	Description									
0	Since this bit was last cleared, no message has been successfully received.									
1	Since this bit was last cleared, a message has been successfully received, independent of the result of the acceptance filtering.									
3	TXOK	R/W	0	<p>Transmitted a Message Successfully</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Since this bit was last cleared, no message has been successfully transmitted.</td></tr> <tr> <td>1</td><td>Since this bit was last cleared, a message has been successfully transmitted error-free and acknowledged by at least one other node.</td></tr> </tbody> </table> <p>This bit must be cleared by writing a 0 to it.</p>	Value	Description	0	Since this bit was last cleared, no message has been successfully transmitted.	1	Since this bit was last cleared, a message has been successfully transmitted error-free and acknowledged by at least one other node.
Value	Description									
0	Since this bit was last cleared, no message has been successfully transmitted.									
1	Since this bit was last cleared, a message has been successfully transmitted error-free and acknowledged by at least one other node.									

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> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>No Error</td></tr> <tr> <td>0x1</td><td>Stuff Error</td></tr> <tr> <td>0x2</td><td>Format Error</td></tr> <tr> <td>0x3</td><td>ACK Error</td></tr> <tr> <td>0x4</td><td>Bit 1 Error</td></tr> <tr> <td>0x5</td><td>Bit 0 Error</td></tr> <tr> <td>0x6</td><td>CRC Error</td></tr> <tr> <td>0x7</td><td>No Event</td></tr> </tbody> </table>	Value	Description	0x0	No Error	0x1	Stuff Error	0x2	Format Error	0x3	ACK Error	0x4	Bit 1 Error	0x5	Bit 0 Error	0x6	CRC Error	0x7	No Event
Value	Description																					
0x0	No Error																					
0x1	Stuff Error																					
0x2	Format Error																					
0x3	ACK Error																					
0x4	Bit 1 Error																					
0x5	Bit 0 Error																					
0x6	CRC Error																					
0x7	No Event																					
				<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. By checking for this status, software can 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 No Event</p> <p>When the LEC bit shows this value, no CAN bus event was detected since this value was written to the LEC field.</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.

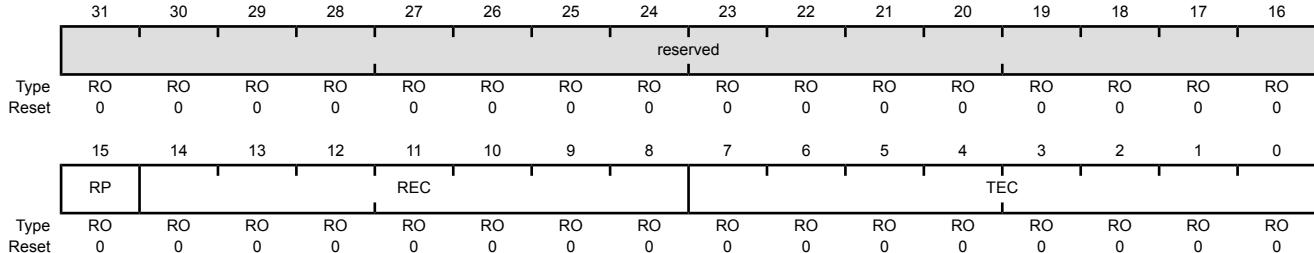
CAN Error Counter (CANERR)

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

Value Description

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
This field contains the state of the receiver error counter (0 to 127).

7:0 TEC RO 0x00 Transmit Error Counter
This field contains the 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 817 for more information.

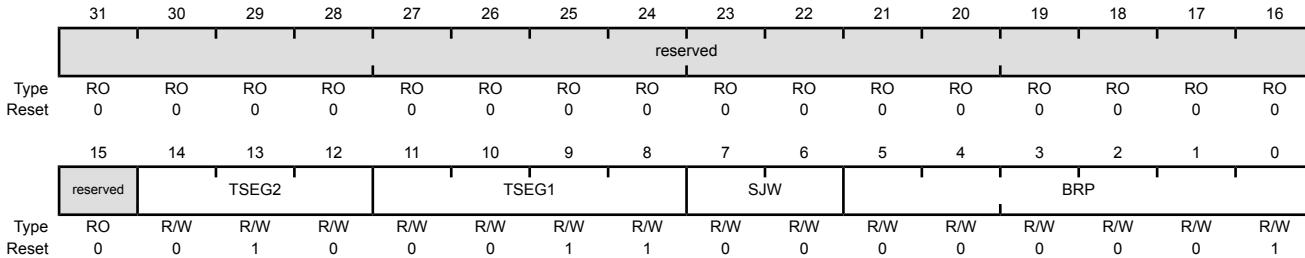
CAN Bit Timing (CANBIT)

CAN0 base: 0x4004.0000

CAN1 base: 0x4004.1000

Offset 0x00C

Type R/W, reset 0x0000.2301



Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14:12	TSEG2	R/W	0x2	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, the reset value of 0x2 means that 3 (2+1) bit time quanta are defined for Phase2 (see Figure 17-4 on page 818). The bit time quanta is defined by the BRP 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 means that 4 (3+1) bit time quanta are defined for Phase1 (see Figure 17-4 on page 818). The bit time quanta is defined by the BRP 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 TSEG2 or TSEG1 by the value in SJW . So the reset value of 0 adjusts the length by 1 bit time quanta.
5:0	BRP	R/W	0x1	Baud Rate Prescaler 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. 0x00-0x03F: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. 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). The CANBRPE register can be used to further divide the bit time.

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.

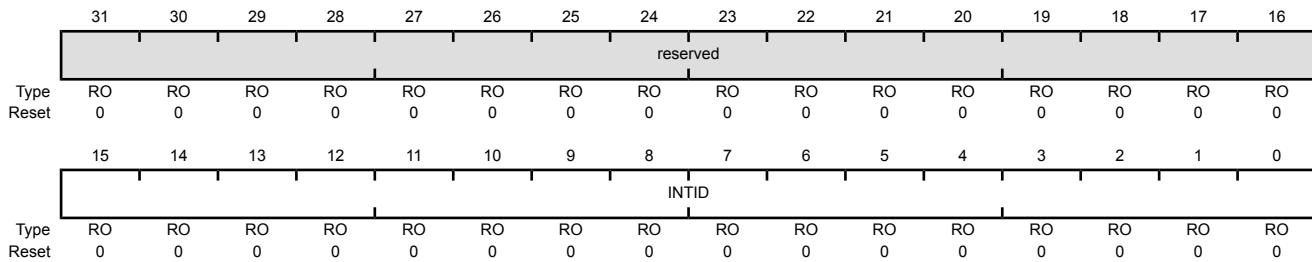
CAN Interrupt (CANINT)

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.

Value	Description
0x0000	No interrupt pending
0x0001-0x0020	Number of the message object that caused the interrupt
0x0021-0x7FFF	Reserved
0x8000	Status Interrupt
0x8001-0xFFFF	Reserved

Register 6: CAN Test (CANTST), offset 0x014

This register is used 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 are affected if the TX bits in this register are not zero.

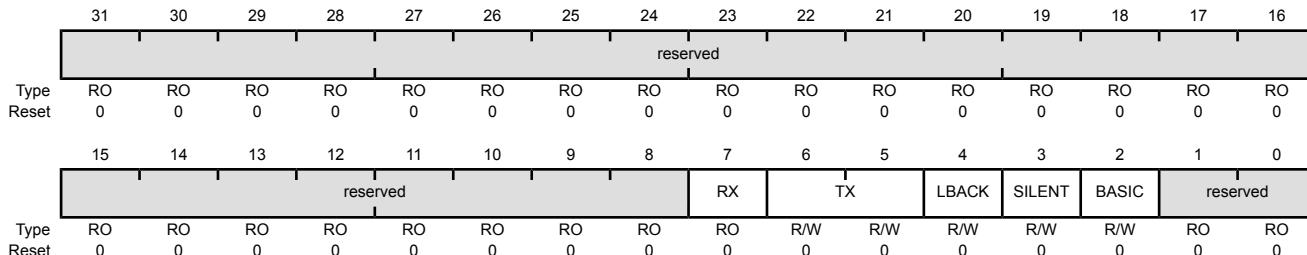
CAN Test (CANTST)

CAN0 base: 0x4004.0000

CAN1 base: 0x4004.1000

Offset 0x014

Type R/W, reset 0x0000.0000



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													
					Value Description													
					0 The CANnRx pin is low.													
					1 The CANnRx pin is high.													
6:5	TX		R/W	0x0	Transmit Control Overrides control of the CANnTx pin.													
					Value Description													
					0x0 CAN Module Control CANnTx is controlled by the CAN module; default operation													
					0x1 Sample Point The sample point is driven on the CANnTx signal. This mode is useful to monitor bit timing.													
					0x2 Driven Low CANnTx drives a low value. This mode is useful for checking the physical layer of the CAN bus.													
					0x3 Driven High CANnTx drives a high value. This mode is useful for checking the physical layer of the CAN bus.													

Bit/Field	Name	Type	Reset	Description						
4	LBACK	R/W	0	Loopback Mode						
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Loopback mode is disabled.</td></tr> <tr> <td>1</td><td>Loopback mode is enabled. In loopback mode, the data from the transmitter is routed into the receiver. Any data on the receive input is ignored.</td></tr> </tbody> </table>	Value	Description	0	Loopback mode is disabled.	1	Loopback mode is enabled. In loopback mode, the data from the transmitter is routed into the receiver. Any data on the receive input is ignored.
Value	Description									
0	Loopback mode is disabled.									
1	Loopback mode is 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						
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Silent mode is disabled.</td></tr> <tr> <td>1</td><td>Silent mode is enabled. In silent mode, the CAN controller does not transmit data but instead monitors the bus. This mode is also known as Bus Monitor mode.</td></tr> </tbody> </table>	Value	Description	0	Silent mode is disabled.	1	Silent mode is enabled. In silent mode, the CAN controller does not transmit data but instead monitors the bus. This mode is also known as Bus Monitor mode.
Value	Description									
0	Silent mode is disabled.									
1	Silent mode is enabled. In silent mode, the CAN controller does not transmit data but instead monitors the bus. This mode is also known as Bus Monitor mode.									
2	BASIC	R/W	0	Basic Mode						
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Basic mode is disabled.</td></tr> <tr> <td>1</td><td>Basic mode is enabled. In basic mode, software should use the CANIF1 registers as the transmit buffer and use the CANIF2 registers as the receive buffer.</td></tr> </tbody> </table>	Value	Description	0	Basic mode is disabled.	1	Basic mode is enabled. In basic mode, software should use the CANIF1 registers as the transmit buffer and use the CANIF2 registers as the receive buffer.
Value	Description									
0	Basic mode is disabled.									
1	Basic mode is enabled. In basic mode, software should use the CANIF1 registers as the transmit buffer and use the CANIF2 registers as the 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.

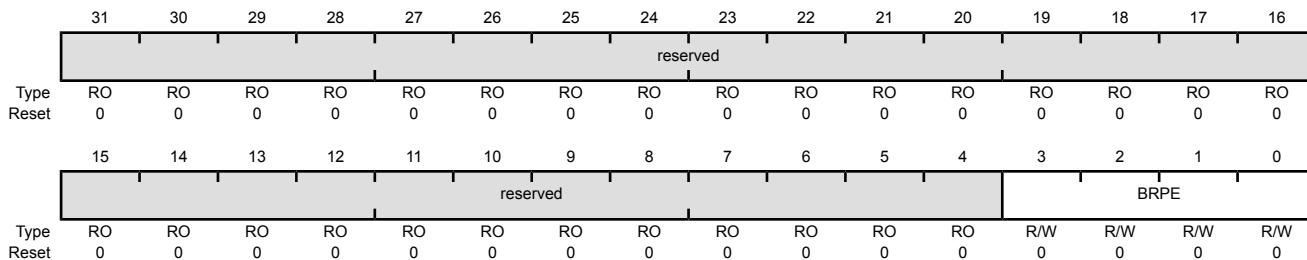
CAN Baud Rate Prescaler Extension (CANBRPE)

CAN0 base: 0x4004.0000

CAN1 base: 0x4004.1000

Offset 0x018

Type R/W, reset 0x0000.0000



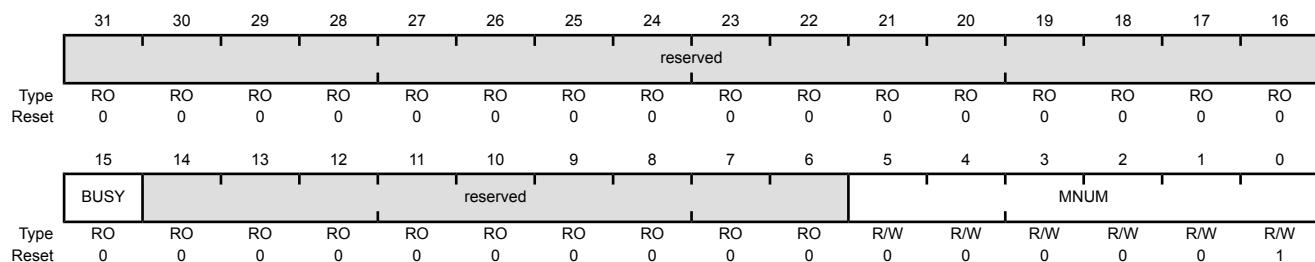
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	BRPE	R/W	0x0	Baud Rate Prescaler Extension 0x00-0x0F: Extend the BRP bit in the CANBIT register to values up to 1023. The actual interpretation by the hardware is one more than the value programmed by BRPE (MSBs) and BRP (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



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
		Value		Description
		0		This bit is cleared when read/write action has finished.
		1		This bit is set when a write occurs to the message number in this register.
14:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	MNUM	R/W	0x01	Message Number Selects one of the 32 message objects in the message RAM for data transfer. The message objects are numbered from 1 to 32.
		Value		Description
		0x00		Reserved 0 is not a valid message number; it is interpreted as 0x20, or object 32.
		0x01-0x20		Message Number Indicates specified message object 1 to 32.
		0x21-0x3F		Reserved Not a valid message number; values are shifted and it is interpreted as 0x01-0x1F.

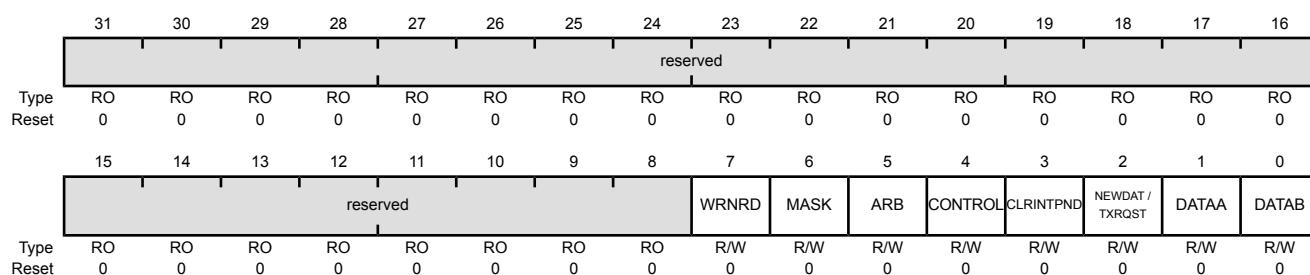
Register 10: CAN IF1 Command Mask (CANIF1CMSK), offset 0x024**Register 11: CAN IF2 Command Mask (CANIF2CMSK), offset 0x084**

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



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
		Value		Description
		0		Transfer the data in the CAN message object specified by the the MNUM field in the CANIFnCRQ register into the CANIFn registers.
		1		Transfer the data in the CANIFn registers to the CAN message object specified by the MNUM field in the CAN Command Request (CANIFnCRQ) .
		Note:		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
		Value		Description
		0		Mask bits unchanged.
		1		Transfer IDMASK + DIR + MXTD of the message object into the Interface registers.

Bit/Field	Name	Type	Reset	Description						
5	ARB	R/W	0	<p>Access Arbitration Bits</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Arbitration bits unchanged.</td></tr> <tr> <td>1</td><td>Transfer ID + DIR + XTD + MSGVAL of the message object into the Interface registers.</td></tr> </tbody> </table>	Value	Description	0	Arbitration bits unchanged.	1	Transfer ID + DIR + XTD + MSGVAL of the message object into the Interface registers.
Value	Description									
0	Arbitration bits unchanged.									
1	Transfer ID + DIR + XTD + MSGVAL of the message object into the Interface registers.									
4	CONTROL	R/W	0	<p>Access Control Bits</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Control bits unchanged.</td></tr> <tr> <td>1</td><td>Transfer control bits from the CANIFnMCTL register into the Interface registers.</td></tr> </tbody> </table>	Value	Description	0	Control bits unchanged.	1	Transfer control bits from the CANIFnMCTL register into the Interface registers.
Value	Description									
0	Control bits unchanged.									
1	Transfer control bits from the CANIFnMCTL register into the Interface registers.									
3	CLRINTPND	R/W	0	<p>Clear Interrupt Pending Bit</p> <p>The function of this bit depends on the configuration of the WRNRD bit.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>If WRNRD is clear, the interrupt pending status is transferred from the message buffer into the CANIFnMCTL register. If WRNRD is set, the INTPND bit in the message object remains unchanged.</td></tr> <tr> <td>1</td><td>If WRNRD is clear, the interrupt pending status is cleared in the message buffer. Note the value of this bit that is transferred to the CANIFnMCTL register always reflects the status of the bits before clearing. If WRNRD is set, the INTPND bit is cleared in the message object.</td></tr> </tbody> </table>	Value	Description	0	If WRNRD is clear, the interrupt pending status is transferred from the message buffer into the CANIFnMCTL register. If WRNRD is set, the INTPND bit in the message object remains unchanged.	1	If WRNRD is clear, the interrupt pending status is cleared in the message buffer. Note the value of this bit that is transferred to the CANIFnMCTL register always reflects the status of the bits before clearing. If WRNRD is set, the INTPND bit is cleared in the message object.
Value	Description									
0	If WRNRD is clear, the interrupt pending status is transferred from the message buffer into the CANIFnMCTL register. If WRNRD is set, the INTPND bit in the message object remains unchanged.									
1	If WRNRD is clear, the interrupt pending status is cleared in the message buffer. Note the value of this bit that is transferred to the CANIFnMCTL register always reflects the status of the bits before clearing. If WRNRD is set, the INTPND bit is cleared in the message object.									
2	NEWDAT / TXRQST	R/W	0	<p>NEWDAT / TXRQST Bit</p> <p>The function of this bit depends on the configuration of the WRNRD bit.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>If WRNRD is clear, the value of the new data status is transferred from the message buffer into the CANIFnMCTL register. If WRNRD is set, a transmission is not requested.</td></tr> <tr> <td>1</td><td>If WRNRD is clear, the new data status is cleared in the message buffer. Note the value of this bit that is transferred to the CANIFnMCTL register always reflects the status of the bits before clearing. If WRNRD is set, a transmission is requested. Note that when this bit is set, the TXRQST bit in the CANIFnMCTL register is ignored.</td></tr> </tbody> </table>	Value	Description	0	If WRNRD is clear, the value of the new data status is transferred from the message buffer into the CANIFnMCTL register. If WRNRD is set, a transmission is not requested.	1	If WRNRD is clear, the new data status is cleared in the message buffer. Note the value of this bit that is transferred to the CANIFnMCTL register always reflects the status of the bits before clearing. If WRNRD is set, a transmission is requested. Note that when this bit is set, the TXRQST bit in the CANIFnMCTL register is ignored.
Value	Description									
0	If WRNRD is clear, the value of the new data status is transferred from the message buffer into the CANIFnMCTL register. If WRNRD is set, a transmission is not requested.									
1	If WRNRD is clear, the new data status is cleared in the message buffer. Note the value of this bit that is transferred to the CANIFnMCTL register always reflects the status of the bits before clearing. If WRNRD is set, a transmission is requested. Note that when this bit is set, the TXRQST bit in the CANIFnMCTL register is ignored.									

Bit/Field	Name	Type	Reset	Description						
1	DATAA	R/W	0	<p>Access Data Byte 0 to 3 The function of this bit depends on the configuration of the WRNRD bit.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Data bytes 0-3 are unchanged.</td></tr> <tr> <td>1</td><td>If WRNRD is clear, transfer data bytes 0-3 in CANIFnDA1 and CANIFnDA2 to the message object. If WRNRD is set, transfer data bytes 0-3 in message object to CANIFnDA1 and CANIFnDA2.</td></tr> </tbody> </table>	Value	Description	0	Data bytes 0-3 are unchanged.	1	If WRNRD is clear, transfer data bytes 0-3 in CANIFnDA1 and CANIFnDA2 to the message object. If WRNRD is set, transfer data bytes 0-3 in message object to CANIFnDA1 and CANIFnDA2 .
Value	Description									
0	Data bytes 0-3 are unchanged.									
1	If WRNRD is clear, transfer data bytes 0-3 in CANIFnDA1 and CANIFnDA2 to the message object. If WRNRD is set, transfer data bytes 0-3 in message object to CANIFnDA1 and CANIFnDA2 .									
0	DATAB	R/W	0	<p>Access Data Byte 4 to 7 The function of this bit depends on the configuration of the WRNRD bit as follows:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Data bytes 4-7 are unchanged.</td></tr> <tr> <td>1</td><td>If WRNRD is clear, transfer data bytes 4-7 in CANIFnDA1 and CANIFnDA2 to the message object. If WRNRD is set, transfer data bytes 4-7 in message object to CANIFnDA1 and CANIFnDA2.</td></tr> </tbody> </table>	Value	Description	0	Data bytes 4-7 are unchanged.	1	If WRNRD is clear, transfer data bytes 4-7 in CANIFnDA1 and CANIFnDA2 to the message object. If WRNRD is set, transfer data bytes 4-7 in message object to CANIFnDA1 and CANIFnDA2 .
Value	Description									
0	Data bytes 4-7 are unchanged.									
1	If WRNRD is clear, transfer data bytes 4-7 in CANIFnDA1 and CANIFnDA2 to the message object. If WRNRD is set, transfer data bytes 4-7 in message object to CANIFnDA1 and CANIFnDA2 .									

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.

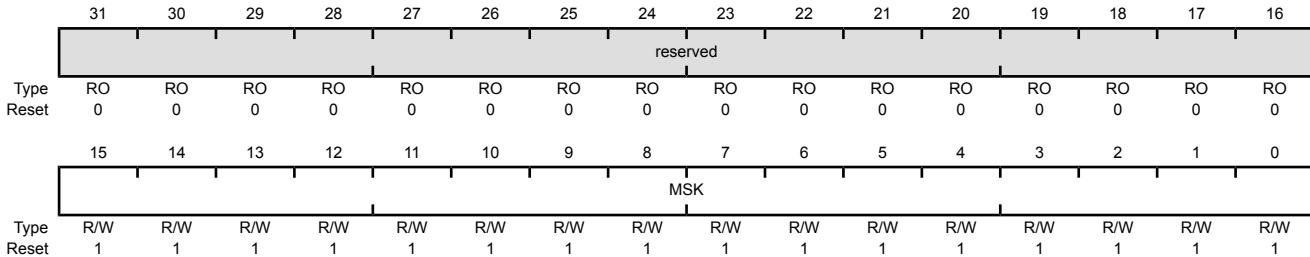
CAN IF1 Mask 1 (CANIF1MSK1)

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 MSK field in the CANIFnMSK2 register are used for bits [28:16] of the ID. When using an 11-bit identifier, these bits are ignored.
Value				Description
0				The corresponding identifier field (ID) in the message object cannot inhibit the match in acceptance filtering.
1				The corresponding identifier field (ID) 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

Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	MXTD	MDIR	reserved												
Reset	R/W	R/W	RO	R/W											

Bit/Field	Name	Type	Reset	Description						
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
15	MXTD	R/W	1	<p>Mask Extended Identifier</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The extended identifier bit (XTD in the CANIFnARB2 register) has no effect on the acceptance filtering.</td> </tr> <tr> <td>1</td> <td>The extended identifier bit XTD is used for acceptance filtering.</td> </tr> </tbody> </table>	Value	Description	0	The extended identifier bit (XTD in the CANIFnARB2 register) has no effect on the acceptance filtering.	1	The extended identifier bit XTD is used for acceptance filtering.
Value	Description									
0	The extended identifier bit (XTD in the CANIFnARB2 register) has no effect on the acceptance filtering.									
1	The extended identifier bit XTD is used for acceptance filtering.									
14	MDIR	R/W	1	<p>Mask Message Direction</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The message direction bit (DIR in the CANIFnARB2 register) has no effect for acceptance filtering.</td> </tr> <tr> <td>1</td> <td>The message direction bit DIR is used for acceptance filtering.</td> </tr> </tbody> </table>	Value	Description	0	The message direction bit (DIR in the CANIFnARB2 register) has no effect for acceptance filtering.	1	The message direction bit DIR is used for acceptance filtering.
Value	Description									
0	The message direction bit (DIR in the CANIFnARB2 register) has no effect for acceptance filtering.									
1	The message direction bit DIR is used for acceptance filtering.									
13	reserved	RO	1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						

Bit/Field	Name	Type	Reset	Description						
12:0	MSK	R/W	0xFF	<p>Identifier Mask</p> <p>When using a 29-bit identifier, these bits are used for bits [28:16] of the ID. The MSK field in the CANIFnMSK1 register are used for bits [15:0] of the ID. When using an 11-bit identifier, MSK[12:2] are used for bits [10:0] of the ID.</p>						
				<table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>The corresponding identifier field (ID) in the message object cannot inhibit the match in acceptance filtering.</td></tr><tr><td>1</td><td>The corresponding identifier field (ID) is used for acceptance filtering.</td></tr></tbody></table>	Value	Description	0	The corresponding identifier field (ID) in the message object cannot inhibit the match in acceptance filtering.	1	The corresponding identifier field (ID) is used for acceptance filtering.
Value	Description									
0	The corresponding identifier field (ID) in the message object cannot inhibit the match in acceptance filtering.									
1	The corresponding identifier field (ID) 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.

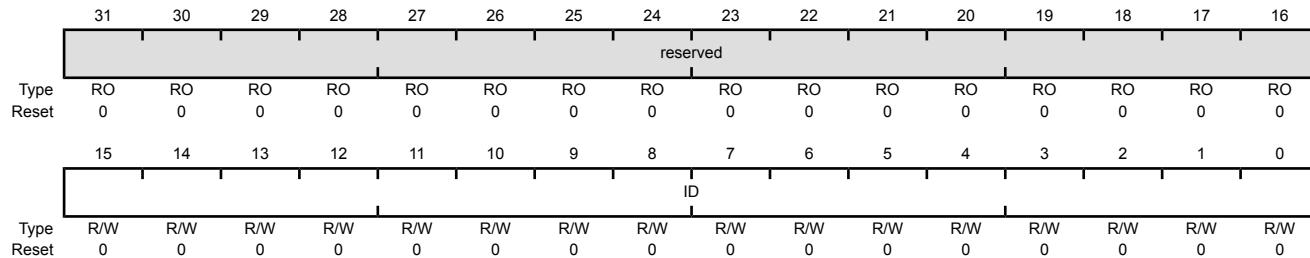
CAN IF1 Arbitration 1 (CANIF1ARB1)

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 ID field in the CANIFnARB2 register to create the message identifier.</p> <p>When using a 29-bit identifier, bits 15:0 of the CANIFnARB1 register are [15:0] of the ID, while bits 12:0 of the CANIFnARB2 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

reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	MSGVAL	XTD	DIR							ID					
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
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	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	Message Valid
				Value Description
			0	The message object is ignored by the message handler.
			1	The message object is configured and ready to be considered by the message handler within the CAN controller.
				All unused message objects should have this bit cleared during initialization and before clearing the INIT bit in the CANCTL register. The MSGVAL bit must also be cleared before any of the following bits are modified or if the message object is no longer required: the ID fields in the CANIFnARBn registers, the XTD and DIR bits in the CANIFnARB2 register, or the DLC field in the CANIFnMCTL register.
14	XTD	R/W	0	Extended Identifier
				Value Description
			0	An 11-bit Standard Identifier is used for this message object.
			1	A 29-bit Extended Identifier is used for this message object.

Bit/Field	Name	Type	Reset	Description						
13	DIR	R/W	0	Message Direction						
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Receive. When the TXRQST bit in the CANIFnMCTL 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.</td></tr> <tr> <td>1</td><td>Transmit. When the TXRQST bit in the CANIFnMCTL register is set, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TXRQST bit of this message object is set (if RMTEN=1).</td></tr> </tbody> </table>	Value	Description	0	Receive. When the TXRQST bit in the CANIFnMCTL 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.	1	Transmit. When the TXRQST bit in the CANIFnMCTL register is set, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TXRQST bit of this message object is set (if RMTEN=1).
Value	Description									
0	Receive. When the TXRQST bit in the CANIFnMCTL 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.									
1	Transmit. When the TXRQST bit in the CANIFnMCTL register is set, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the TXRQST bit of this message object is set (if RMTEN=1).									
12:0	ID	R/W	0x000	<p>Message Identifier</p> <p>This bit field is used with the ID field in the CANIFnARB2 register to create the message identifier.</p> <p>When using a 29-bit identifier, ID[15:0] of the CANIFnARB1 register are [15:0] of the ID, while these bits, ID[12:0], are [28:16] of the ID. When using an 11-bit identifier, ID[12:2] are used for bits [10:0] of the ID. The ID field in the CANIFnARB1 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

reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	NEWDAT	MSGLST	INTPND	UMASK	TXIE	RXIE	RMTEN	TXRQST	EOB	reserved			DLC		
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	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
		Value		Description
	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
		Value		Description
	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 CANIFnARB2 register is clear (receive).			
13	INTPND	R/W	0	Interrupt Pending
		Value		Description
	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 CANINT register points to this message object if there is not another interrupt source with a higher priority.		

Bit/Field	Name	Type	Reset	Description						
12	UMASK	R/W	0	Use Acceptance Mask						
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Mask is ignored.</td></tr> <tr> <td>1</td><td>Use mask (MSK, MXTD, and MDIR bits in the CANIFnMSKn registers) for acceptance filtering.</td></tr> </tbody> </table>	Value	Description	0	Mask is ignored.	1	Use mask (MSK, MXTD, and MDIR bits in the CANIFnMSKn registers) for acceptance filtering.
Value	Description									
0	Mask is ignored.									
1	Use mask (MSK, MXTD, and MDIR bits in the CANIFnMSKn registers) for acceptance filtering.									
11	TXIE	R/W	0	Transmit Interrupt Enable						
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The INTPND bit in the CANIFnMCTL register is unchanged after a successful transmission of a frame.</td></tr> <tr> <td>1</td><td>The INTPND bit in the CANIFnMCTL register is set after a successful transmission of a frame.</td></tr> </tbody> </table>	Value	Description	0	The INTPND bit in the CANIFnMCTL register is unchanged after a successful transmission of a frame.	1	The INTPND bit in the CANIFnMCTL register is set after a successful transmission of a frame.
Value	Description									
0	The INTPND bit in the CANIFnMCTL register is unchanged after a successful transmission of a frame.									
1	The INTPND bit in the CANIFnMCTL register is set after a successful transmission of a frame.									
10	RXIE	R/W	0	Receive Interrupt Enable						
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The INTPND bit in the CANIFnMCTL register is unchanged after a successful reception of a frame.</td></tr> <tr> <td>1</td><td>The INTPND bit in the CANIFnMCTL register is set after a successful reception of a frame.</td></tr> </tbody> </table>	Value	Description	0	The INTPND bit in the CANIFnMCTL register is unchanged after a successful reception of a frame.	1	The INTPND bit in the CANIFnMCTL register is set after a successful reception of a frame.
Value	Description									
0	The INTPND bit in the CANIFnMCTL register is unchanged after a successful reception of a frame.									
1	The INTPND bit in the CANIFnMCTL register is set after a successful reception of a frame.									
9	RMTEN	R/W	0	Remote Enable						
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>At the reception of a remote frame, the TXRQST bit in the CANIFnMCTL register is left unchanged.</td></tr> <tr> <td>1</td><td>At the reception of a remote frame, the TXRQST bit in the CANIFnMCTL register is set.</td></tr> </tbody> </table>	Value	Description	0	At the reception of a remote frame, the TXRQST bit in the CANIFnMCTL register is left unchanged.	1	At the reception of a remote frame, the TXRQST bit in the CANIFnMCTL register is set.
Value	Description									
0	At the reception of a remote frame, the TXRQST bit in the CANIFnMCTL register is left unchanged.									
1	At the reception of a remote frame, the TXRQST bit in the CANIFnMCTL register is set.									
8	TXRQST	R/W	0	Transmit Request						
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>This message object is not waiting for transmission.</td></tr> <tr> <td>1</td><td>The transmission of this message object is requested and is not yet done.</td></tr> </tbody> </table> <p>Note: If the WRNRD and TXRQST bits in the CANIFnCMSK register are set, this bit is ignored.</p>	Value	Description	0	This message object is not waiting for transmission.	1	The transmission of this message object is requested and is not yet done.
Value	Description									
0	This message object is not waiting for transmission.									
1	The transmission of this message object is requested and is not yet done.									

Bit/Field	Name	Type	Reset	Description						
7	EOB	R/W	0	End of Buffer						
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Message object belongs to a FIFO Buffer and is not the last message object of that FIFO Buffer.</td></tr> <tr> <td>1</td><td>Single message object or last message object of a FIFO Buffer.</td></tr> </tbody> </table>	Value	Description	0	Message object belongs to a FIFO Buffer and is not the last message object of that FIFO Buffer.	1	Single message object or last message object of a FIFO Buffer.
Value	Description									
0	Message object belongs to a FIFO Buffer and is not the last message object of that FIFO Buffer.									
1	Single message object or last message object of a FIFO Buffer.									
				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.						
6:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
3:0	DLC	R/W	0x0	Data Length Code						
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0-0x8</td><td>Specifies the number of bytes in the data frame.</td></tr> <tr> <td>0x9-0xF</td><td>Defaults to a data frame with 8 bytes.</td></tr> </tbody> </table>	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.									
				The DLC field in the CANIFnMCTL 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 DLC to the value given by the received message.						

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.

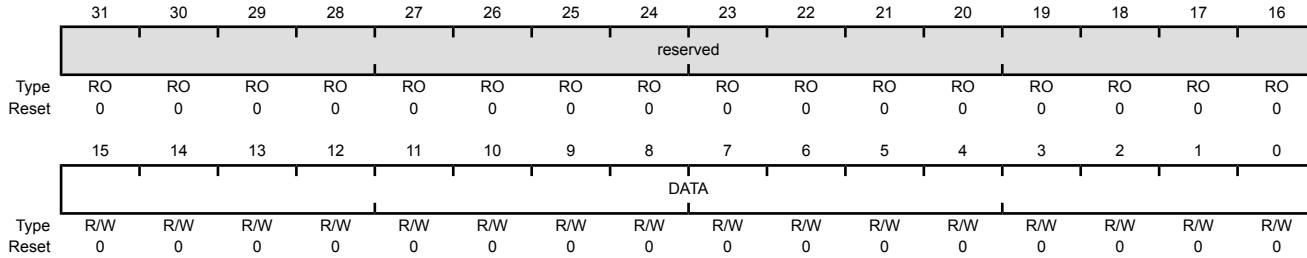
CAN IF1 Data A1 (CANIF1DA1)

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.

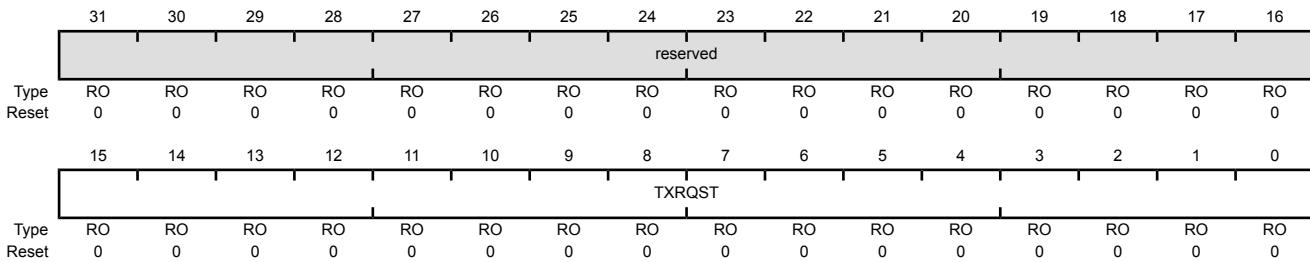
The **CANTXRQ1** register contains the TXRQST bits of the first 16 message objects in the message RAM; the **CANTXRQ2** register contains the TXRQST bits of the second 16 message objects.

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
		Value		Description
		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.

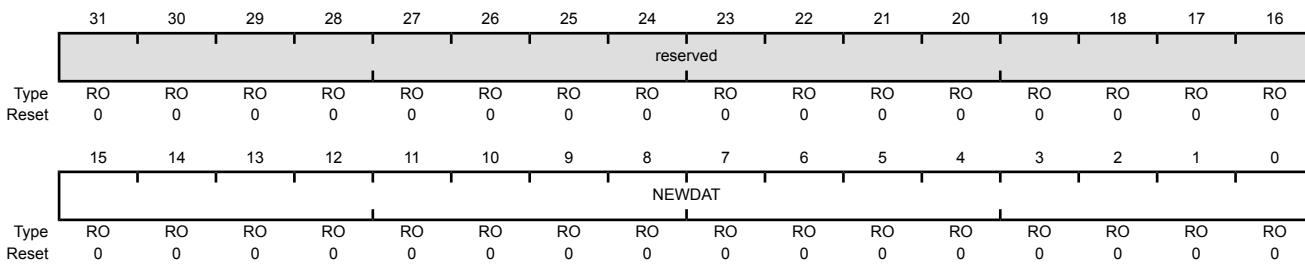
CAN New Data 1 (CANNWDA1)

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
		Value		Description
		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

reserved															
Type	RO														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
INTPND															
Type	RO														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	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
		Value		Description
		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 valid bit of a specific message object can be changed with the **CANIFnARB2** 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.

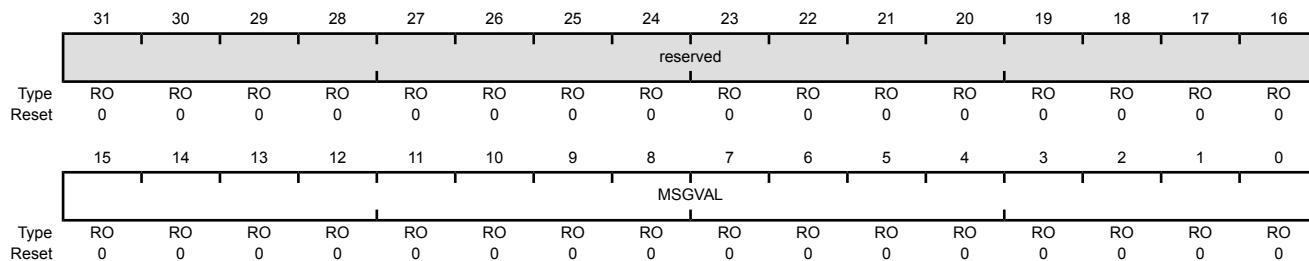
CAN Message 1 Valid (CANMSG1VAL)

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

Value	Description
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.

18 Ethernet Controller

The Stellaris® Ethernet Controller consists of a fully integrated media access controller (MAC) and network physical (PHY) interface. The Ethernet Controller conforms to *IEEE 802.3* specifications and fully supports 10BASE-T and 100BASE-TX standards.

The Stellaris Ethernet Controller module has the following features:

- Conforms to the *IEEE 802.3-2002 specification*
 - 10BASE-T/100BASE-TX IEEE-802.3 compliant. Requires only a dual 1:1 isolation transformer interface to the line
 - 10BASE-T/100BASE-TX ENDEC, 100BASE-TX scrambler/descrambler
 - Full-featured auto-negotiation
- Multiple operational modes
 - Full- and half-duplex 100 Mbps
 - Full- and half-duplex 10 Mbps
 - Power-saving and power-down modes
- Highly configurable
 - Programmable MAC address
 - LED activity selection
 - Promiscuous mode support
 - CRC error-rejection control
 - User-configurable interrupts
- Physical media manipulation
 - MDI/MDI-X cross-over support through software assist
 - Register-programmable transmit amplitude
 - Automatic polarity correction and 10BASE-T signal reception
- IEEE 1588 Precision Time Protocol: Provides highly accurate time stamps for individual packets
- Efficient transfers using Micro Direct Memory Access Controller (μ DMA)
 - Separate channels for transmit and receive
 - Receive channel request asserted on packet receipt
 - Transmit channel request asserted on empty transmit FIFO

18.1 Block Diagram

As shown in Figure 18-1 on page 854, the Ethernet Controller is functionally divided into two layers: the Media Access Controller (MAC) layer and the Network Physical (PHY) layer. These layers correspond to the OSI model layers 2 and 1, respectively. The CPU accesses the Ethernet Controller via the MAC layer. The MAC layer provides transmit and receive processing for Ethernet frames. The MAC layer also provides the interface to the PHY layer via an internal Media Independent Interface (MII). The PHY layer communicates with the Ethernet bus.

Figure 18-1. Ethernet Controller

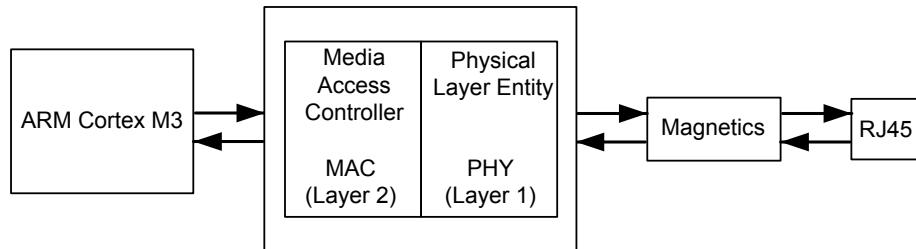
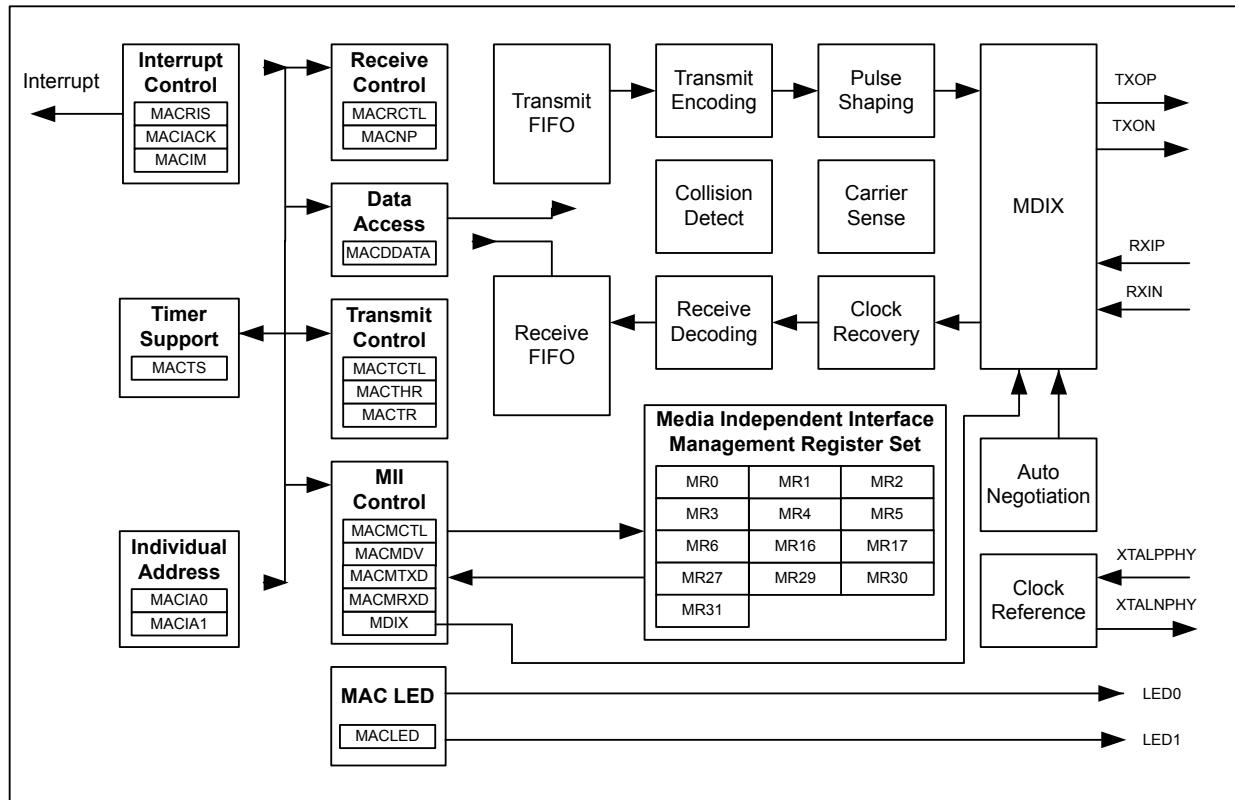


Figure 18-2 on page 854 shows more detail of the internal structure of the Ethernet Controller and how the register set relates to various functions.

Figure 18-2. Ethernet Controller Block Diagram



18.2 Signal Description

Table 18-1 on page 855 and Table 18-2 on page 855 list the external signals of the Ethernet Controller and describe the function of each. The Ethernet LED signals are alternate functions for GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the LED signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 447) should be set to choose the LED function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 465) to assign the LED0 and LED1 signals to the specified GPIO port pins. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 423. The remaining signals (with the word "fixed" in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

Table 18-1. Signals for Ethernet (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
ERBIAS	33	fixed	O	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
LED0	59	PF3 (1)	O	TTL	Ethernet LED 0.
LED1	60	PF2 (1)	O	TTL	Ethernet LED 1.
MDIO	58	fixed	I/O	OD	MDIO of the Ethernet PHY.
RXIN	37	fixed	I	Analog	RXIN of the Ethernet PHY.
RXIP	40	fixed	I	Analog	RXIP of the Ethernet PHY.
TXON	46	fixed	O	TTL	TXON of the Ethernet PHY.
TXOP	43	fixed	O	TTL	TXOP of the Ethernet PHY.
XTALNPHY	17	fixed	O	Analog	Ethernet PHY XTALN 25-MHz oscillator crystal output.
XTALPPHY	16	fixed	I	Analog	Ethernet PHY XTALP 25-MHz oscillator crystal input.

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

Table 18-2. Signals for Ethernet (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
ERBIAS	J3	fixed	O	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
LED0	J12	PF3 (1)	O	TTL	Ethernet LED 0.
LED1	J11	PF2 (1)	O	TTL	Ethernet LED 1.
MDIO	L9	fixed	I/O	OD	MDIO of the Ethernet PHY.
RXIN	L7	fixed	I	Analog	RXIN of the Ethernet PHY.
RXIP	M7	fixed	I	Analog	RXIP of the Ethernet PHY.
TXON	L8	fixed	O	TTL	TXON of the Ethernet PHY.
TXOP	M8	fixed	O	TTL	TXOP of the Ethernet PHY.
XTALNPHY	J1	fixed	O	Analog	Ethernet PHY XTALN 25-MHz oscillator crystal output.
XTALPPHY	J2	fixed	I	Analog	Ethernet PHY XTALP 25-MHz oscillator crystal input.

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

18.3 Functional Description

Note: A 12.4-k Ω resistor should be connected between the ERBIAS and ground. The 12.4-k Ω resistor should have a 1% tolerance and should be located in close proximity to the ERBIAS pin. Power dissipation in the resistor is low, so a chip resistor of any geometry may be used.

The functional description of the Ethernet Controller is discussed in the following sections.

18.3.1 MAC Operation

The following sections describe the operation of the MAC layer, including an overview of the Ethernet frame format, the MAC layer FIFOs, Ethernet transmission and reception options, packet timestamps, and LED indicators.

18.3.1.1 Ethernet Frame Format

Ethernet data is carried by Ethernet frames. The basic frame format is shown in Figure 18-3 on page 856.

Figure 18-3. Ethernet Frame

Preamble	SFD	Destination Address	Source Address	Length/ Type	Data	FCS
7 Bytes	1 Byte	6 Bytes	6 Bytes	2 Bytes	46 - 1500 Bytes	4 Bytes

The seven fields of the frame are transmitted from left to right. The bits within the frame are transmitted from least to most significant bit.

- Preamble

The Preamble field is used to synchronize with the received frame's timing. The preamble is 7 octets long.

- Start Frame Delimiter (SFD)

The SFD field follows the preamble pattern and indicates the start of the frame. Its value is 1010.1011b.

- Destination Address (DA)

This field specifies destination addresses for which the frame is intended. The LSB (bit 16 of DA oct 1 in the frame, see Table 18-3 on page 858) of the DA determines whether the address is an individual (0), or group/multicast (1) address.

- Source Address (SA)

The source address field identifies the station from which the frame was initiated.

- Length/Type Field

The meaning of this field depends on its numeric value. This field can be interpreted as length or type code. The maximum length of the data field is 1500 octets. If the value of the Length/Type field is less than or equal to 1500 decimal, it indicates the number of MAC client data octets. If the value of this field is greater than or equal to 1536 decimal, then it encodes the type interpretation. The meaning of the Length/Type field when the value is between 1500 and 1536 decimal is unspecified by the IEEE 802.3 standard. However, the Ethernet Controller assumes

type interpretation if the value of the Length/Type field is greater than 1500 decimal. The definition of the Type field is specified in the IEEE 802.3 standard. The first of the two octets in this field is most significant.

- Data

The data field is a sequence of octets that is at least 46 in length, up to 1500 in length. Full data transparency is provided so any values can appear in this field. A minimum frame size of 46 octets is required to meet the IEEE standard. If the frame size is too small, the Ethernet Controller automatically appends extra bits (a pad), thus the pad can have a size of 0 to 46 octets. Data padding can be disabled by clearing the **PADEN** bit in the **Ethernet MAC Transmit Control (MACTCTL)** register.

For the Ethernet Controller, data sent/received can be larger than 1500 bytes without causing a Frame Too Long error. Instead, a FIFO overrun error is reported using the **FOV** bit in the **Ethernet MAC Raw Interrupt Status (MACRIS)** register when the frame received is too large to fit into the Ethernet Controller's 2K RAM.

- Frame Check Sequence (FCS)

The frame check sequence carries the cyclic redundancy check (CRC) value. The CRC is computed over the destination address, source address, length/type, and data (including pad) fields using the CRC-32 algorithm. The Ethernet Controller computes the FCS value one nibble at a time. For transmitted frames, this field is automatically inserted by the MAC layer, unless disabled by clearing the **CRC** bit in the **MACTCTL** register. For received frames, this field is automatically checked. If the FCS does not pass, the frame is not placed in the RX FIFO, unless the FCS check is disabled by clearing the **BADCRC** bit in the **MACRCTL** register.

18.3.1.2 MAC Layer FIFOs

The Ethernet Controller is capable of simultaneous transmission and reception. This feature is enabled by setting the **DUPLEX** bit in the **MACTCTL** register.

For Ethernet frame transmission, a 2-KB transmit FIFO is provided that can be used to store a single frame. While the *IEEE 802.3 specification* limits the size of an Ethernet frame's payload section to 1500 Bytes, the Ethernet Controller places no such limit. The full buffer can be used for a payload of up to 2032 bytes (as the first 16 bytes in the FIFO are reserved for destination address, source address and length/type information).

For Ethernet frame reception, a 2-KB receive FIFO is provided that can be used to store multiple frames, up to a maximum of 31 frames. If a frame is received, and there is insufficient space in the RX FIFO, an overflow error is indicated using the **FOV** bit in the **MACRIS** register.

For details regarding the TX and RX FIFO layout, refer to Table 18-3 on page 858. Please note the following difference between TX and RX FIFO layout. For the TX FIFO, the Data Length field in the first FIFO word refers to the Ethernet frame data payload, as shown in the 5th to nth FIFO positions. For the RX FIFO, the Frame Length field is the total length of the received Ethernet frame, including the Length/Type bytes and the FCS bits.

If FCS generation is disabled by clearing the **CRC** bit in the **MACTCTL** register, the last word in the TX FIFO must contain the FCS bytes for the frame that has been written to the FIFO.

Also note that if the length of the data payload section is not a multiple of 4, the FCS field is not be aligned on a word boundary in the FIFO. However, for the RX FIFO, the beginning of the next frame is always on a word boundary.

Table 18-3. TX & RX FIFO Organization

FIFO Word Read/Write Sequence	Word Bit Fields	TX FIFO (Write)	RX FIFO (Read)
1st	7:0	Data Length Least Significant Byte	Frame Length Least Significant Byte
	15:8	Data Length Most Significant Byte	Frame Length Most Significant Byte
	23:16		DA oct 1
	31:24		DA oct 2
2nd	7:0		DA oct 3
	15:8		DA oct 4
	23:16		DA oct 5
	31:24		DA oct 6
3rd	7:0		SA oct 1
	15:8		SA oct 2
	23:16		SA oct 3
	31:24		SA oct 4
4th	7:0		SA oct 5
	15:8		SA oct 6
	23:16		Len/Type Most Significant Byte
	31:24		Len/Type Least Significant Byte
5th to nth	7:0		data oct n
	15:8		data oct n+1
	23:16		data oct n+2
	31:24		data oct n+3
last	7:0		FCS 1 ^a
	15:8		FCS 2 ^a
	23:16		FCS 3 ^a
	31:24		FCS 4 ^a

a. If the CRC bit in the **MACTCTL** register is clear, the FCS bytes must be written with the correct CRC. If the CRC bit is set, the Ethernet Controller automatically writes the FCS bytes.

18.3.1.3 Ethernet Transmission Options

At the MAC layer, the transmitter can be configured for both full-duplex and half-duplex operation by using the DUPLEX bit in the **MACTCTL** register. Note that in 10BASE-T half-duplex mode, the transmitted data is looped back on the receive path.

The Ethernet Controller automatically generates and inserts the Frame Check Sequence (FCS) at the end of the transmit frame when the CRC bit in the **MACTCTL** register is set. However, for test purposes, this feature can be disabled in order to generate a frame with an invalid CRC by clearing the CRC bit.

The *IEEE 802.3 specification* requires that the Ethernet frame payload section be a minimum of 46 bytes. The Ethernet Controller automatically pads the data section if the payload data section loaded into the FIFO is less than the minimum 46 bytes when the PADEN bit in the **MACTCTL** register is set. This feature can be disabled by clearing the PADEN bit.

The transmitter must be enabled by setting the TXEN bit in the **MACTCTL** register.

18.3.1.4 Ethernet Reception Options

The Ethernet Controller RX FIFO should be cleared during software initialization. The receiver should first be disabled by clearing the RXEN bit in the **Ethernet MAC Receive Control (MACRCTL)** register, then the FIFO can be cleared by setting the RSTFIFO bit in the **MACRCTL** register.

The receiver automatically rejects frames that contain bad CRC values in the FCS field. In this case, a Receive Error interrupt is generated and the receive data is lost. To accept all frames, clear the BADCRC bit in the **MACRCTL** register.

In normal operating mode, the receiver accepts only those frames that have a destination address that matches the address programmed into the **Ethernet MAC Individual Address 0 (MACIA0)** and **Ethernet MAC Individual Address 1 (MACIA1)** registers. However, the Ethernet receiver can also be configured for Promiscuous and Multicast modes by setting the PRMS and AMUL bits in the **MACRCTL** register.

18.3.1.5 Packet Timestamps

Some applications require a very precise clock for time stamping samples or triggering events. The IEEE Precision Time Protocol (PTP), or IEEE-1588, provides a mechanism for synchronizing clocks across an Ethernet to sub-microsecond precision. The accuracy of the PTP clock depends greatly upon the accuracy of timestamps of the PTP Ethernet packets. In a software-only PTP solution, there can be jitter in the Ethernet packet timestamps, resulting in a less precise PTP clock on the target. In some Stellaris devices, General-Purpose Timer 3 (GPT3) can be used in conjunction with the **Ethernet MAC Timer Support (MACTS)** register to provide a more accurate timestamp for Ethernet packets.

This feature is enabled by setting the TSEN bit in the **MACTS** register. Note that when this feature is enabled, GPT3 must be dedicated to the Ethernet Controller. GPT3 must be configured to 16-bit edge capture mode, see page 496. Timer A of GPT3 stores the transmit time, and Timer B stores the receive time. One other General-Purpose Timer can be set up as a 16-bit free-running timer to synchronize the receiver and transmitter timers and provide a timestamp with which to compare the timestamps stored in GPT3. The enet_ptpd example in the StellarisWare® software package provides a sample PTP application that illustrates both software-only time stamping as well the use of the GPT3 and **MACTS** register for more accurate timestamps. This example supports version 1 of the IEEE-1588 protocol, but Stellaris microcontrollers support both versions 1 and 2.

18.3.1.6 LED Indicators

The Ethernet Controller supports two LED signals that can be used to indicate various states of operation. These signals are mapped to the LED0 and LED1 pins. By default, these pins are configured as GPIO signals (PF3 and PF2). For the Ethernet Controller to drive these signals, they must be reconfigured to their hardware function. See “General-Purpose Input/Outputs (GPIOs)” on page 423 for additional details. The function of these pins is programmable using the **Ethernet MAC LED Encoding (MACLED)** register. Refer to page 891 for additional details on how to program these LED functions.

18.3.2 Internal MII Operation

For the MII management interface to function properly, the MDIO signal must be connected through a 10 kΩ pull-up resistor to the +3.3 V supply. Failure to connect this pull-up resistor prevents management transactions on this internal MII to function. Note that it is possible for data transmission across the MII to still function since the PHY layer auto-negotiates the link parameters by default.

For the MII management interface to function properly, the internal clock must be divided down from the system clock to a frequency no greater than 2.5 MHz. The **Ethernet MAC Management Divider**

(**MACMDV**) register contains the divider used for scaling down the system clock. See page 885 for more details about the use of this register.

18.3.3 PHY Operation

The Physical Layer (PHY) in the Ethernet Controller includes integrated ENDECs, scrambler/descrambler, dual-speed clock recovery, and full-featured auto-negotiation functions. The transmitter includes an on-chip pulse shaper and a low-power line driver. The receiver has an adaptive equalizer and a baseline restoration circuit required for accurate clock and data recovery. The transceiver interfaces to Category-5 unshielded twisted pair (Cat-5 UTP) cabling for 100BASE-TX applications, and Category-3 unshielded twisted pair (Cat-3 UTP) for 10BASE-T applications. The Ethernet Controller is connected to the line media via dual 1:1 isolation transformers. No external filter is required.

18.3.3.1 Clock Selection

The Ethernet Controller can be clocked from an on-chip crystal oscillator which can also be driven by an external oscillator. When using the on-chip crystal oscillator, a 25-MHz crystal should be connected between the **XTALPPHY** and **XTALNPHY** pins. Alternatively, an external 25-MHz clock input can be connected to the **XTALPPHY** pin. In this mode of operation, a crystal is not required and the **XTALNPHY** pin should be left unconnected. See “Ethernet Controller” on page 1249 for more information regarding the specifications of the Ethernet Controller.

18.3.3.2 Auto-Negotiation

The Ethernet Controller supports the auto-negotiation functions of Clause 28 of the *IEEE 802.3* standard for 10/100 Mbps operation over copper wiring. This function is controlled via register settings. The auto-negotiation function is turned on by default, and the **ANEGEN** bit in the **Ethernet PHY Management Register 0 - Control (MR0)** is set after reset. Software can disable the auto-negotiation function by clearing the **ANEGEN** bit. The contents of the **Ethernet PHY Management Register - Auto-Negotiation Advertisement (MR4)** are reflected to the Ethernet Controller’s link partner during auto-negotiation via fast-link pulse coding.

Once auto-negotiation is complete, the **SPEED** bit in the **Ethernet PHY Management Register 31 – PHY Special Control/Status (MR31)** register reflects the actual speed. The **AUTODONE** bit in **MR31** is set to indicate that auto-negotiation is complete. Setting the **RANEG** bit in the **MR0** register also causes auto-negotiation to restart.

18.3.3.3 Polarity Correction

The Ethernet Controller is capable of automatic polarity reversal for 10BASE-T and auto-negotiation functions. The **XPOL** bit in the **Ethernet PHY Management Register 27 –Special Control/Status (MR27)** register is set to indicate the polarity has automatically been reversed.

18.3.3.4 MDI/MDI-X Configuration

The Ethernet Controller supports the MDI/MDI-X configuration as defined in *IEEE 802.3-2002 specification* through software assistance. The MDI/MDI-X configuration eliminates the need for cross-over cables when connecting to another device, such as a hub. Software can implement the MDI/MDI-X configuration using a function outlined by the pseudo code below. This code should be called periodically using one of the available timer resources on the Stellaris microcontroller such as the System Tick Timer or one of the General Purpose timers. The following code refers to the **LINK** bit in the **Ethernet PHY Management Register 1 - Status (MR1)**, the **ENON** bit in the **Ethernet PHY Management Register 17 - Mode Control/Status (MR17)**, and the **EN** bit of the **Ethernet PHY MDIX (MDIX)** register.

```

//
// Entry Point for MDI/MDI-X configuration.
//

//
// Increment the Link Active and Energy Detect Timers using the elapsed time
// since the last call to this function. If using a periodic timer, the
// elapsed time should be a constant (the programmed period of the timer).
//
Increment Link Active Timer
Increment Energy Detect Timer

//
if(No Ethernet Link Active)
{
//
// If energy has been detected on the link, reset the Energy Detect Timer.
// If it is a "new" energy detect, reset the link detect timer also.
//
if(Ethernet Energy Detected)
{
    Reset Energy Detect Timer

    if(New Energy Detect)
    {
        Reset Link Detect Timer
    }
}

//
// If the Energy or Link Detect timer has expired, toggle the MDI/MDI-X
// mode. Typically, the Energy Detect Timer would be ~62ms, while the
// Link Detect Timer would be ~2s
//
if((Energy Detect Timer Expired) or
    (Link Detect Timer Expired))
{
    Reset Energy Detect Timer

    if(Random Event)
    {
        Reset Link Detect Timer
        Toggle MDI/MDI-X Mode
    }
}

//
// Here, if an Ethernet Link has been detected, simply reset the timers
// for the next time around.
//
else

```

```
{  
    Reset Link Detect Timer  
    Reset Energy Detect Timer  
}
```

18.3.3.5 Power Management

The PHY has two power-saving modes:

- Power-Down
- Energy Detect Power-Down

Power-down mode is activated by setting the **PWRDN** bit in the **MR0** register. When the PHY is in power-down mode, it consumes minimum power. When the **PWRDN** bit is cleared, the PHY powers up and is automatically reset.

The energy detect power-down mode is activated by setting the **EDPD** bit in the **MR17** register. In this mode of operation, when no energy is present on the line, the PHY is powered down, except for the management interface, the SQUELCH circuit and the ENERGYON logic. The ENERGYON logic is used to detect the presence of valid energy from 100BASE-T, 10BASE-T, or auto-negotiation signals. While the PHY is powered down, nothing is transmitted. When link pulses or packets are received, the PHY powers-up. The PHY automatically resets itself into the state it had prior to power down and sets the **EONIS** bit in the **MR29** register. The first and possibly the second packet to activate the ENERGYON mode may be lost.

18.3.4 Interrupts

The Ethernet Controller can generate an interrupt for one or more of the following conditions:

- A frame has been received into an empty RX FIFO
- A frame transmission error has occurred
- A frame has been transmitted successfully
- A frame has been received with inadequate room in the RX FIFO (overrun)
- A frame has been received with one or more error conditions (for example, FCS failed)
- An MII management transaction between the MAC and PHY layers has completed
- One or more of the following PHY layer conditions occurs:
 - Auto-Negotiate Complete
 - Remote Fault
 - Link Partner Acknowledge
 - Parallel Detect Fault
 - Page Received

Refer to **Ethernet PHY Management Register 29 - Interrupt Source Flags (MR29)** (see page 909) for additional details regarding PHY interrupts.

18.3.5 DMA Operation

The Ethernet peripheral provides request signals to the µDMA controller and has a dedicated channel for transmit and one for receive. The request is a single type for both channels. Burst requests are not supported. The RX channel request is asserted when a packet is received while the TX channel request is asserted when the transmit FIFO becomes empty.

No special configuration is needed to enable the Ethernet peripheral for use with the µDMA controller.

Because the size of a received packet is not known until the header is examined, it is best to set up the initial µDMA transfer to copy the first 4 words including the packet length plus the Ethernet header from the RX FIFO when the RX request occurs. The µDMA causes an interrupt when this transfer is complete. Upon entering the interrupt handler, the packet length in the FIFO and the Ethernet header are in a buffer and can be examined. Once the packet length is known, then another µDMA transfer can be set up to transfer the remaining received packet payload from the FIFO into a buffer. This transfer should be initiated by software. Another interrupt occurs when this transfer is done.

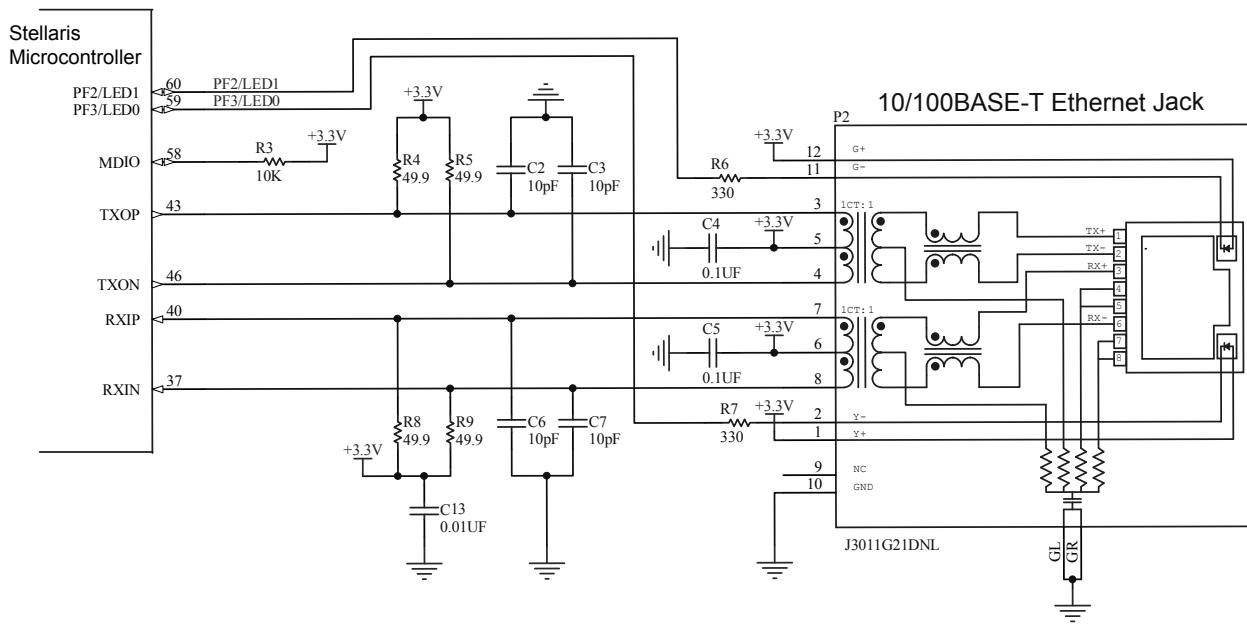
Even though the TX channel generates a TX empty request, the recommended way to handle µDMA transfers for transmitting packets is to set up the transfer from the buffer containing the packet to the transmit FIFO, and then to initiate the transfer with a software request. An interrupt occurs when this transfer is complete. For both channels, the "auto-request" transfer mode should be used. See "Micro Direct Memory Access (µDMA)" on page 365 for more details about programming the µDMA controller.

18.4 Initialization and Configuration

The following sections describe the hardware and software configuration required to set up the Ethernet Controller.

18.4.1 Hardware Configuration

Figure 18-4 on page 864 shows the proper method for interfacing the Ethernet Controller to a 10/100BASE-T Ethernet jack.

Figure 18-4. Interface to an Ethernet Jack

The following isolation transformers have been tested and are known to successfully interface to the Ethernet PHY layer.

- Isolation Transformers
 - TDK TLA-6T103
 - TDK TLA-6T118
 - Bel-Fuse S558-5999-46
 - Halo TG22-3506ND
 - Halo TG110-S050
 - PCA EPF8023G
 - Pulse PE-68515
 - Valor ST6118
 - YCL 20PMT04
- Isolation transformers with integrated RJ45 connector
 - TDK TLA-6T704
 - Delta RJS-1A08T089A
- Isolation transformers with integrated RJ45 connector, LEDs and termination resistors
 - Pulse J0011D21B/E
 - Pulse J3011G21DNL

18.4.2 Software Configuration

To use the Ethernet Controller, it must be enabled by setting the EPHY0 and EMAC0 bits in the **RCGC2** register (see page 285). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module. See page 285. To find out which GPIO port to enable, refer to Table 24-4 on page 1182. Configure the **PMCn** fields in the **GPIOCTL** register to assign the Ethernet signals to the appropriate pins. See page 465 and Table 24-5 on page 1190.

The following steps can then be used to configure the Ethernet Controller for basic operation.

1. Program the **MACDIV** register to obtain a 2.5 MHz clock (or less) on the internal MII. Assuming a 20-MHz system clock, the **MACDIV** value should be 0x03 or greater.
2. Program the **MACIA0** and **MACIA1** register for address filtering.
3. Program the **MACTCTL** register for Auto CRC generation, padding, and full-duplex operation using a value of 0x16.
4. Program the **MACRCTL** register to flush the receive FIFO and reject frames with bad FCS using a value of 0x18.
5. Enable both the Transmitter and Receive by setting the LSB in both the **MACTCTL** and **MACRCTL** registers.
6. To transmit a frame, write the frame into the TX FIFO using the **Ethernet MAC Data (MACDATA)** register. Then set the **NEWTX** bit in the **Ethernet Mac Transmission Request (MACTR)** register to initiate the transmit process. When the **NEWTX** bit has been cleared, the TX FIFO is available for the next transmit frame.
7. To receive a frame, wait for the **NPR** field in the **Ethernet MAC Number of Packets (MACNP)** register to be non-zero. Then begin reading the frame from the RX FIFO by using the **MACDATA** register. To ensure that the entire packet is received, either use the DriverLib EthernetPacketGet() API or compare the number of bytes received to the Length field from the frame to determine when the packet has been completely read.

18.5 Register Map

Table 18-4 on page 865 lists the Ethernet MAC and MII Management registers. The MAC register addresses given are relative to the Ethernet base address of 0x4004.8000. The MII Management registers are accessed using the **MACMCTL** register. Note that the Ethernet controller clocks must be enabled before the registers can be programmed (see page 285).

The *IEEE 802.3* standard specifies a register set for controlling and gathering status from the PHY layer. The registers are collectively known as the MII Management registers and are detailed in Section 22.2.4 of the *IEEE 802.3 specification*. Table 18-4 on page 865 also lists these MII Management registers. All addresses given are absolute and are written directly to the *REGADR* field of the **Ethernet MAC Management Control (MACMCTL)** register. The format of registers 0 to 15 are defined by the IEEE specification and are common to all PHY layer implementations. The only variance allowed is for features that may or may not be supported by a specific PHY implementation. Registers 16 to 31 are vendor-specific registers, used to support features that are specific to a vendor's PHY implementation.

Table 18-4. Ethernet Register Map

Offset	Name	Type	Reset	Description	See page
Ethernet MAC (Ethernet Offset)					
0x000	MACRIS/MACIACK	R/W1C	0x0000.0000	Ethernet MAC Raw Interrupt Status/Acknowledge	868
0x004	MACIM	R/W	0x0000.007F	Ethernet MAC Interrupt Mask	871
0x008	MACRCTL	R/W	0x0000.0008	Ethernet MAC Receive Control	873
0x00C	MACTCTL	R/W	0x0000.0000	Ethernet MAC Transmit Control	875

Table 18-4. Ethernet Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x010	MACDATA	R/W	0x0000.0000	Ethernet MAC Data	877
0x014	MACIA0	R/W	0x0000.0000	Ethernet MAC Individual Address 0	879
0x018	MACIA1	R/W	0x0000.0000	Ethernet MAC Individual Address 1	880
0x01C	MACTHR	R/W	0x0000.003F	Ethernet MAC Threshold	881
0x020	MACMCTL	R/W	0x0000.0000	Ethernet MAC Management Control	883
0x024	MACMDV	R/W	0x0000.0080	Ethernet MAC Management Divider	885
0x02C	MACMTXD	R/W	0x0000.0000	Ethernet MAC Management Transmit Data	886
0x030	MACMRXD	R/W	0x0000.0000	Ethernet MAC Management Receive Data	887
0x034	MACNP	RO	0x0000.0000	Ethernet MAC Number of Packets	888
0x038	MACTR	R/W	0x0000.0000	Ethernet MAC Transmission Request	889
0x03C	MACTS	R/W	0x0000.0000	Ethernet MAC Timer Support	890
0x040	MACLED	R/W	0x0000.0100	Ethernet MAC LED Encoding	891
0x044	MDIX	R/W	0x0000.0000	Ethernet PHY MDIX	893

MII Management (Accessed through the MACMCTL register)

-	MR0	R/W	0x1000	Ethernet PHY Management Register 0 – Control	894
-	MR1	RO	0x7809	Ethernet PHY Management Register 1 – Status	896
-	MR2	RO	0x0161	Ethernet PHY Management Register 2 – PHY Identifier 1	898
-	MR3	RO	0xB410	Ethernet PHY Management Register 3 – PHY Identifier 2	899
-	MR4	R/W	0x01E1	Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement	900
-	MR5	RO	0x0001	Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability	902
-	MR6	RO	0x0000	Ethernet PHY Management Register 6 – Auto-Negotiation Expansion	904
-	MR16	RO	0x0040	Ethernet PHY Management Register 16 – Vendor-Specific	905
-	MR17	R/W	0x0002	Ethernet PHY Management Register 17 – Mode Control/Status	906
-	MR27	RO	-	Ethernet PHY Management Register 27 – Special Control/Status	908
-	MR29	RO	0x0000	Ethernet PHY Management Register 29 – Interrupt Status	909
-	MR30	R/W	0x0000	Ethernet PHY Management Register 30 – Interrupt Mask	911
-	MR31	R/W	0x0040	Ethernet PHY Management Register 31 – PHY Special Control/Status	913

18.6 Ethernet MAC Register Descriptions

The remainder of this section lists and describes the Ethernet MAC registers, in numerical order by address offset. Also see “MII Management Register Descriptions” on page 893.

Register 1: Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK), offset 0x000

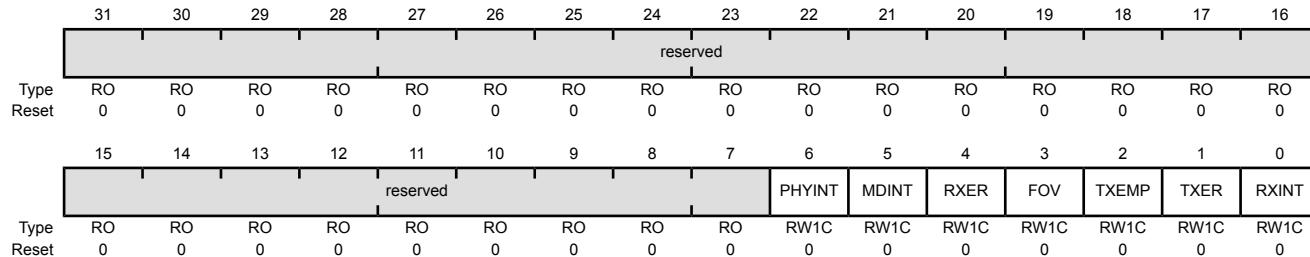
The **MACRIS/MACIACK** register is the interrupt status and acknowledge register. On a read, this register gives the current status value of the corresponding interrupt prior to masking. On a write, setting any bit clears the corresponding interrupt status bit.

Ethernet MAC Raw Interrupt Status/Acknowledge (MACRIS/MACIACK)

Base 0x4004.8000

Offset 0x000

Type R/W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:7	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.
6	PHYINT	RW1C	0	PHY Interrupt
		Value	Description	
		1	An enabled interrupt in the PHY layer has occurred. MR29 in the PHY must be read to determine the specific PHY event that triggered this interrupt.	
		0	No interrupt.	
		This bit is cleared by writing a 1 to it.		
5	MDINT	RW1C	0	MII Transaction Complete
		Value	Description	
		1	A transaction (read or write) on the MII interface has completed successfully.	
		0	No interrupt.	
		This bit is cleared by writing a 1 to it.		

Bit/Field	Name	Type	Reset	Description
4	RXER	RW1C	0	<p>Receive Error</p> <p>Value Description</p> <p>1 An error was encountered on the receiver. The possible errors that can cause this interrupt bit to be set are:</p> <ul style="list-style-type: none"> ■ A receive error occurs during the reception of a frame (100 Mbps only). ■ The frame is not an integer number of bytes (dribble bits) due to an alignment error. ■ The CRC of the frame does not pass the FCS check. ■ The length/type field is inconsistent with the frame data size when interpreted as a length field. <p>0 No interrupt.</p> <p>This bit is cleared by writing a 1 to it.</p>
3	FOV	RW1C	0	<p>FIFO Overrun</p> <p>Value Description</p> <p>1 An overrun was encountered on the receive FIFO.</p> <p>0 No interrupt.</p> <p>This bit is cleared by writing a 1 to it.</p>
2	TXEMP	RW1C	0	<p>Transmit FIFO Empty</p> <p>Value Description</p> <p>1 The packet was transmitted and that the TX FIFO is empty.</p> <p>0 No interrupt.</p> <p>This bit is cleared by writing a 1 to it.</p>
1	TXER	RW1C	0	<p>Transmit Error</p> <p>Value Description</p> <p>1 An error was encountered on the transmitter. The possible errors that can cause this interrupt bit to be set are:</p> <ul style="list-style-type: none"> ■ The data length field stored in the TX FIFO exceeds 2032 decimal (buffer length - 16 bytes of header data). The frame is not sent when this error occurs. ■ The retransmission attempts during the backoff process have exceeded the maximum limit of 16 decimal. <p>0 No interrupt.</p> <p>Writing a 1 to this bit clears it and resets the TX FIFO write pointer.</p>

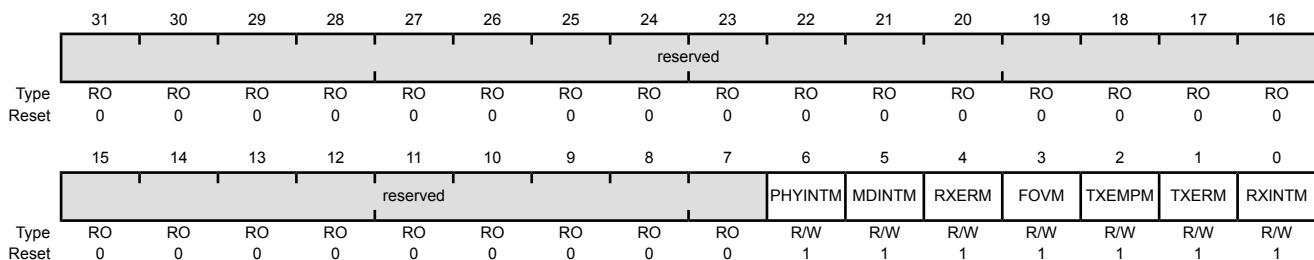
Bit/Field	Name	Type	Reset	Description
0	RXINT	RW1C	0	Packet Received
Value Description				
		1		At least one packet has been received and is stored in the receiver FIFO.
		0		No interrupt.
This bit is cleared by writing a 1 to it.				

Register 2: Ethernet MAC Interrupt Mask (MACIM), offset 0x004

This register allows software to enable/disable Ethernet MAC interrupts. Clearing a bit disables the interrupt, while setting the bit enables it.

Ethernet MAC Interrupt Mask (MACIM)

Base 0x4004.8000
Offset 0x004
Type R/W, reset 0x0000.007F



Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	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.
6	PHYINTM	R/W	1	Mask PHY Interrupt
	Value Description			
	1	An interrupt is sent to the interrupt controller when the PHYINT bit in the MACRIS/MACIACK register is set.		
	0	The PHYINT interrupt is suppressed and not sent to the interrupt controller.		
5	MDINTM	R/W	1	Mask MII Transaction Complete
	Value Description			
	1	An interrupt is sent to the interrupt controller when the MDINT bit in the MACRIS/MACIACK register is set.		
	0	The MDINT interrupt is suppressed and not sent to the interrupt controller.		
4	RXERM	R/W	1	Mask Receive Error
	Value Description			
	1	An interrupt is sent to the interrupt controller when the RXER bit in the MACRIS/MACIACK register is set.		
	0	The RXER interrupt is suppressed and not sent to the interrupt controller.		

Bit/Field	Name	Type	Reset	Description
3	FOVM	R/W	1	Mask FIFO Overrun Value Description 1 An interrupt is sent to the interrupt controller when the FOV bit in the MACRIS/MACIACK register is set. 0 The FOV interrupt is suppressed and not sent to the interrupt controller.
2	TXEMPM	R/W	1	Mask Transmit FIFO Empty Value Description 1 An interrupt is sent to the interrupt controller when the TXEMP bit in the MACRIS/MACIACK register is set. 0 The TXEMP interrupt is suppressed and not sent to the interrupt controller.
1	TXERM	R/W	1	Mask Transmit Error Value Description 1 An interrupt is sent to the interrupt controller when the TXER bit in the MACRIS/MACIACK register is set. 0 The TXER interrupt is suppressed and not sent to the interrupt controller.
0	RXINTM	R/W	1	Mask Packet Received Value Description 1 An interrupt is sent to the interrupt controller when the RXINT bit in the MACRIS/MACIACK register is set. 0 The RXINT interrupt is suppressed and not sent to the interrupt controller.

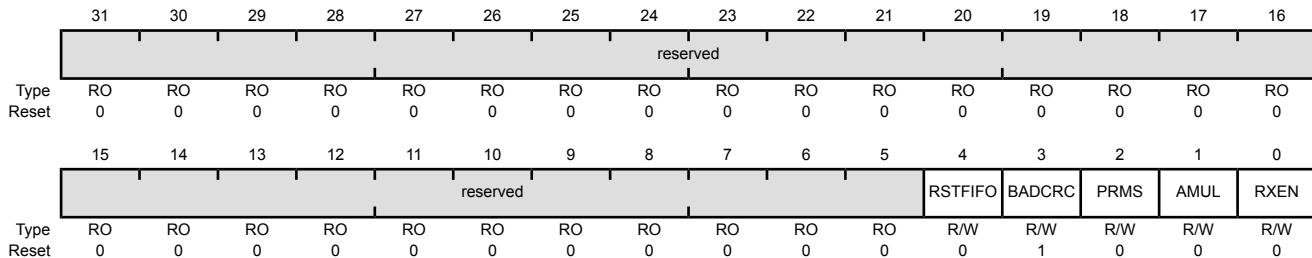
Register 3: Ethernet MAC Receive Control (MACRCTL), offset 0x008

This register configures the receiver and controls the types of frames that are received.

It is important to note that when the receiver is enabled, all valid frames with a broadcast address of FF-FF-FF-FF-FF-FF in the Destination Address field are received and stored in the RX FIFO, even if the AMUL bit is not set.

Ethernet MAC Receive Control (MACRCTL)

Base 0x4004.8000
Offset 0x008
Type R/W, reset 0x0000.0008



Bit/Field	Name	Type	Reset	Description						
31:5	reserved	RO	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	RSTFIFO	R/W	0	Clear Receive FIFO <table border="0"> <tr> <th>Value</th> <th>Description</th> </tr> <tr> <td>1</td> <td>Clear the receive FIFO. The receive FIFO should be cleared when software initialization is performed.</td> </tr> <tr> <td>0</td> <td>No effect.</td> </tr> </table> This bit is automatically cleared when read. The receiver should be disabled (RXEN = 0), before a reset is initiated (RSTFIFO = 1). This sequence flushes and resets the RX FIFO.	Value	Description	1	Clear the receive FIFO. The receive FIFO should be cleared when software initialization is performed.	0	No effect.
Value	Description									
1	Clear the receive FIFO. The receive FIFO should be cleared when software initialization is performed.									
0	No effect.									
3	BADCRC	R/W	1	Enable Reject Bad CRC <table border="0"> <tr> <th>Value</th> <th>Description</th> </tr> <tr> <td>1</td> <td>Enables the rejection of frames with an incorrectly calculated CRC. If a bad CRC is encountered, the RXER bit in the MACRIS register is set and the receiver FIFO is reset.</td> </tr> <tr> <td>0</td> <td>Disables the rejection of frames with an incorrectly calculated CRC.</td> </tr> </table>	Value	Description	1	Enables the rejection of frames with an incorrectly calculated CRC. If a bad CRC is encountered, the RXER bit in the MACRIS register is set and the receiver FIFO is reset.	0	Disables the rejection of frames with an incorrectly calculated CRC.
Value	Description									
1	Enables the rejection of frames with an incorrectly calculated CRC. If a bad CRC is encountered, the RXER bit in the MACRIS register is set and the receiver FIFO is reset.									
0	Disables the rejection of frames with an incorrectly calculated CRC.									
2	PRMS	R/W	0	Enable Promiscuous Mode <table border="0"> <tr> <th>Value</th> <th>Description</th> </tr> <tr> <td>1</td> <td>Enables Promiscuous mode, which accepts all valid frames, regardless of the specified Destination Address.</td> </tr> <tr> <td>0</td> <td>Disables Promiscuous mode, accepting only frames with the programmed Destination Address.</td> </tr> </table>	Value	Description	1	Enables Promiscuous mode, which accepts all valid frames, regardless of the specified Destination Address.	0	Disables Promiscuous mode, accepting only frames with the programmed Destination Address.
Value	Description									
1	Enables Promiscuous mode, which accepts all valid frames, regardless of the specified Destination Address.									
0	Disables Promiscuous mode, accepting only frames with the programmed Destination Address.									

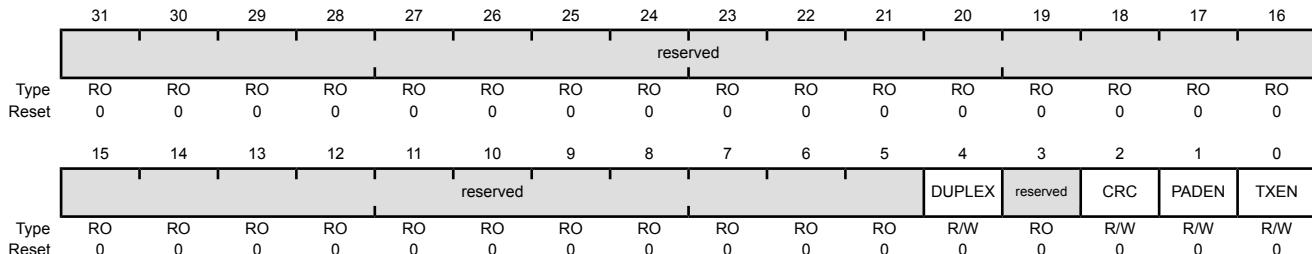
Bit/Field	Name	Type	Reset	Description
1	AMUL	R/W	0	Enable Multicast Frames Value Description 1 Enables the reception of multicast frames. 0 Disables the reception of multicast frames.
0	RXEN	R/W	0	Enable Receiver Value Description 1 Enables the Ethernet receiver. 0 Disables the receiver. All frames are ignored.

Register 4: Ethernet MAC Transmit Control (MACTCTL), offset 0x00C

This register configures the transmitter and controls the frames that are transmitted.

Ethernet MAC Transmit Control (MACTCTL)

Base 0x4004.8000
Offset 0x00C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description						
31:5	reserved	RO	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	DUPLEX	R/W	0	Enable Duplex Mode <table border="0" style="margin-left: 20px;"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>1</td> <td>Enables Duplex mode, allowing simultaneous transmission and reception.</td> </tr> <tr> <td>0</td> <td>Disables Duplex mode.</td> </tr> </table>	Value	Description	1	Enables Duplex mode, allowing simultaneous transmission and reception.	0	Disables Duplex mode.
Value	Description									
1	Enables Duplex mode, allowing simultaneous transmission and reception.									
0	Disables Duplex mode.									
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	CRC	R/W	0	Enable CRC Generation <table border="0" style="margin-left: 20px;"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>1</td> <td>Enables the automatic generation of the CRC and its placement at the end of the packet.</td> </tr> <tr> <td>0</td> <td>The frames placed in the TX FIFO are sent exactly as they are written into the FIFO.</td> </tr> </table> Note that this bit should generally be set.	Value	Description	1	Enables the automatic generation of the CRC and its placement at the end of the packet.	0	The frames placed in the TX FIFO are sent exactly as they are written into the FIFO.
Value	Description									
1	Enables the automatic generation of the CRC and its placement at the end of the packet.									
0	The frames placed in the TX FIFO are sent exactly as they are written into the FIFO.									
1	PADEN	R/W	0	Enable Packet Padding <table border="0" style="margin-left: 20px;"> <tr> <td>Value</td> <td>Description</td> </tr> <tr> <td>1</td> <td>Enables the automatic padding of packets that do not meet the minimum frame size.</td> </tr> <tr> <td>0</td> <td>Disables automatic padding.</td> </tr> </table> Note that this bit should generally be set.	Value	Description	1	Enables the automatic padding of packets that do not meet the minimum frame size.	0	Disables automatic padding.
Value	Description									
1	Enables the automatic padding of packets that do not meet the minimum frame size.									
0	Disables automatic padding.									

Bit/Field	Name	Type	Reset	Description
0	TXEN	R/W	0	Enable Transmitter
Value Description				
		1		Enables the transmitter.
		0		Disables the transmitter.

Register 5: Ethernet MAC Data (MACDATA), offset 0x010

Important: Use caution when reading this register. Performing a read may change bit status.

This register enables software to access the TX and RX FIFOs.

Reads from this register return the data stored in the RX FIFO from the location indicated by the read pointer. The read pointer is then auto incremented to the next RX FIFO location. Reading from the RX FIFO when a frame has not been received or is in the process of being received returns indeterminate data and does not increment the read pointer.

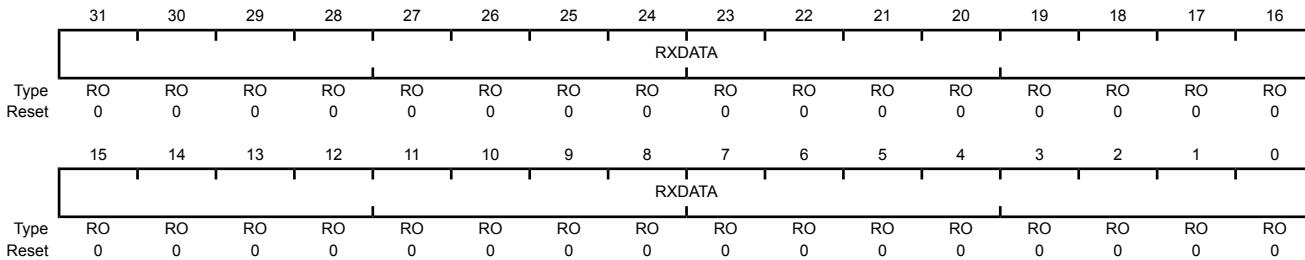
Writes to this register store the data in the TX FIFO at the location indicated by the write pointer. The write pointer is then auto incremented to the next TX FIFO location. Writing more data into the TX FIFO than indicated in the length field results in the data being lost. Writing less data into the TX FIFO than indicated in the length field results in indeterminate data being appended to the end of the frame to achieve the indicated length. Attempting to write the next frame into the TX FIFO before transmission of the first has completed results in the data being lost.

Bytes may not be randomly accessed in either the RX or TX FIFOs. Data must be read from the RX FIFO sequentially and stored in a buffer for further processing. Once a read has been performed, the data in the FIFO cannot be re-read. Data must be written to the TX FIFO sequentially. If an error is made in placing the frame into the TX FIFO, the write pointer can be reset to the start of the TX FIFO by writing the TXER bit of the **MACIACK** register and then the data re-written.

Reads

Ethernet MAC Data (MACDATA)

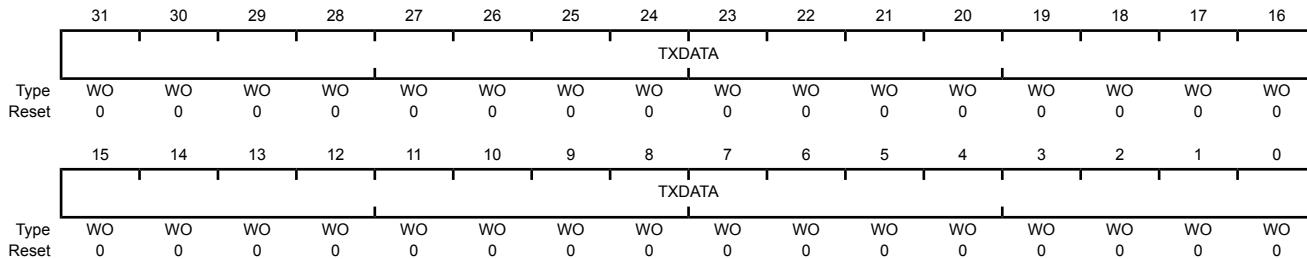
Base 0x4004.8000
Offset 0x010
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	RXDATA	RO	0x0000.0000	Receive FIFO Data The RXDATA bits represent the next word of data stored in the RX FIFO.

Writes**Ethernet MAC Data (MACDATA)**

Base 0x4004.8000
 Offset 0x010
 Type WO, reset 0x0000.0000



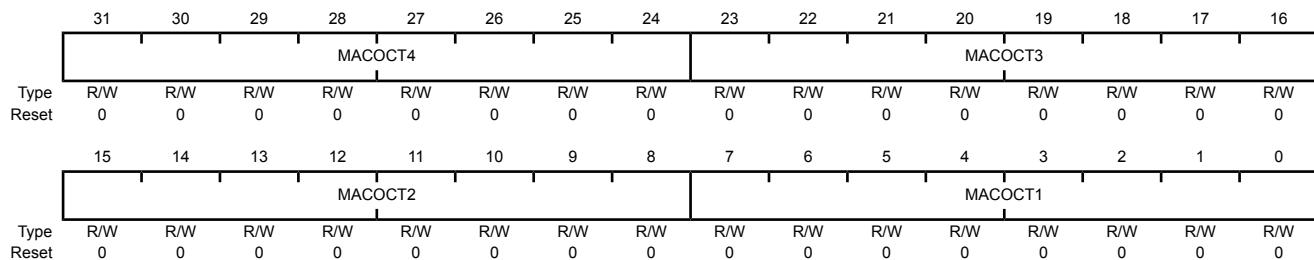
Bit/Field	Name	Type	Reset	Description
31:0	TXDATA	WO	0x0000.0000	Transmit FIFO Data The TXDATA bits represent the next word of data to place in the TX FIFO for transmission.

Register 6: Ethernet MAC Individual Address 0 (MACIA0), offset 0x014

This register enables software to program the first four bytes of the hardware MAC address of the Network Interface Card (NIC). The last two bytes are in **MACIA1**. The 6-byte Individual Address is compared against the incoming Destination Address fields to determine whether the frame should be received.

Ethernet MAC Individual Address 0 (MACIA0)

Base 0x4004.8000
Offset 0x014
Type R/W, reset 0x0000.0000



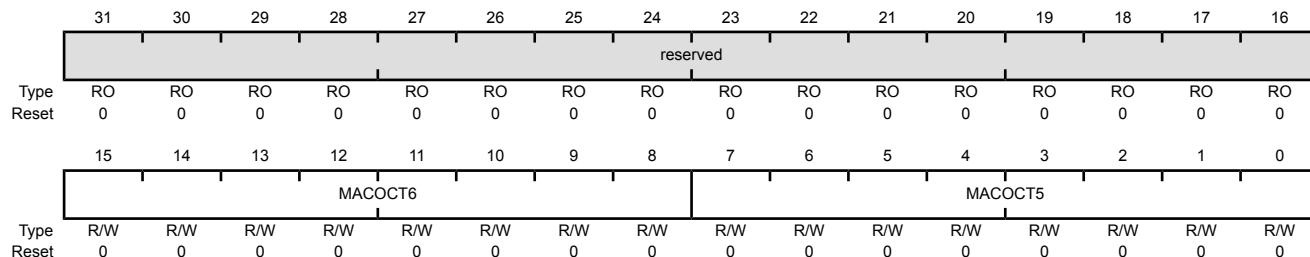
Bit/Field	Name	Type	Reset	Description
31:24	MACOCT4	R/W	0x00	MAC Address Octet 4 The MACOCT4 bits represent the fourth octet of the MAC address used to uniquely identify the Ethernet Controller.
23:16	MACOCT3	R/W	0x00	MAC Address Octet 3 The MACOCT3 bits represent the third octet of the MAC address used to uniquely identify the Ethernet Controller.
15:8	MACOCT2	R/W	0x00	MAC Address Octet 2 The MACOCT2 bits represent the second octet of the MAC address used to uniquely identify the Ethernet Controller.
7:0	MACOCT1	R/W	0x00	MAC Address Octet 1 The MACOCT1 bits represent the first octet of the MAC address used to uniquely identify the Ethernet Controller.

Register 7: Ethernet MAC Individual Address 1 (MACIA1), offset 0x018

This register enables software to program the last two bytes of the hardware MAC address of the Network Interface Card (NIC). The first four bytes are in **MACIA0**. The 6-byte IAR is compared against the incoming Destination Address fields to determine whether the frame should be received.

Ethernet MAC Individual Address 1 (MACIA1)

Base 0x4004.8000
Offset 0x018
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	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.
15:8	MACOCT6	R/W	0x00	MAC Address Octet 6 The MACOCT6 bits represent the sixth octet of the MAC address used to uniquely identify each Ethernet Controller.
7:0	MACOCT5	R/W	0x00	MAC Address Octet 5 The MACOCT5 bits represent the fifth octet of the MAC address used to uniquely identify the Ethernet Controller.

Register 8: Ethernet MAC Threshold (MACTHR), offset 0x01C

In order to increase the transmission rate, it is possible to program the Ethernet Controller to begin transmission of the next frame prior to the completion of the transmission of the current frame.

Caution – Extreme care must be used when implementing this function. Software must be able to guarantee that the complete frame is able to be stored in the transmission FIFO prior to the completion of the transmission frame.

This register enables software to set the threshold level at which the transmission of the frame begins. If the THRESH bits are set to 0x3F, which is the reset value, the early transmission feature is disabled, and transmission does not start until the NEWTX bit is set in the MACTR register.

Writing the THRESH field to any value besides 0x3F enables the early transmission feature. Once the byte count of data in the TX FIFO reaches the value derived from the THRESH bits as shown below, transmission of the frame begins. When the THRESH field is clear, transmission of the frame begins after 4 bytes (a single write) are stored in the TX FIFO. Each increment of the THRESH bit field waits for an additional 32 bytes of data (eight writes) to be stored in the TX FIFO. Therefore, a value of 0x01 causes the transmitter to wait for 36 bytes of data to be written while a value of 0x02 makes the wait equal to 68 bytes of written data. In general, early transmission starts when:

$$\text{Number of Bytes} \geq 4 ((\text{THRESH} \times 8) + 1)$$

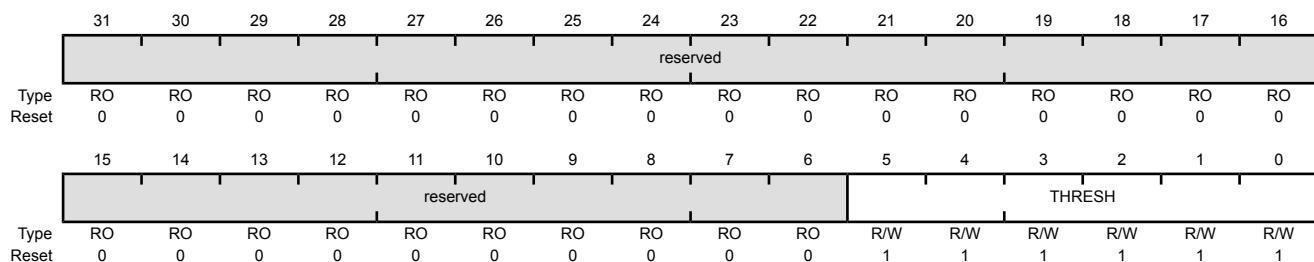
Reaching the threshold level has the same effect as setting the NEWTX bit in the MACTR register. Transmission of the frame begins, and then the number of bytes indicated by the Data Length field is transmitted. Because underrun checking is not performed, if any event, such as an interrupt, delays the filling of the FIFO, the tail pointer may reach and pass the write pointer in the TX FIFO. In this event, indeterminate values are transmitted rather than the end of the frame. Therefore, sufficient bus bandwidth for writing to the TX FIFO must be guaranteed by the software.

If a frame smaller than the threshold level must be sent, the NEWTX bit in the MACTR register must be set with an explicit write, which initiates the transmission of the frame even though the threshold limit has not been reached.

If the threshold level is set too small, it is possible for the transmitter to underrun. If this occurs, the transmit frame is aborted, and a transmit error occurs. Note that in this case, the TXER bit in the MACRIS is not set, meaning that the CPU receives no indication that a transmit error happened.

Ethernet MAC Threshold (MACTHR)

Base 0x4004.8000
Offset 0x01C
Type R/W, reset 0x0000.003F



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	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
5:0	THRESH	R/W	0x3F	<p>Threshold Value</p> <p>The THRESH bits represent the early transmit threshold. Once the amount of data in the TX FIFO exceeds the value represented by the above equation, transmission of the packet begins.</p>

Register 9: Ethernet MAC Management Control (MACMCTL), offset 0x020

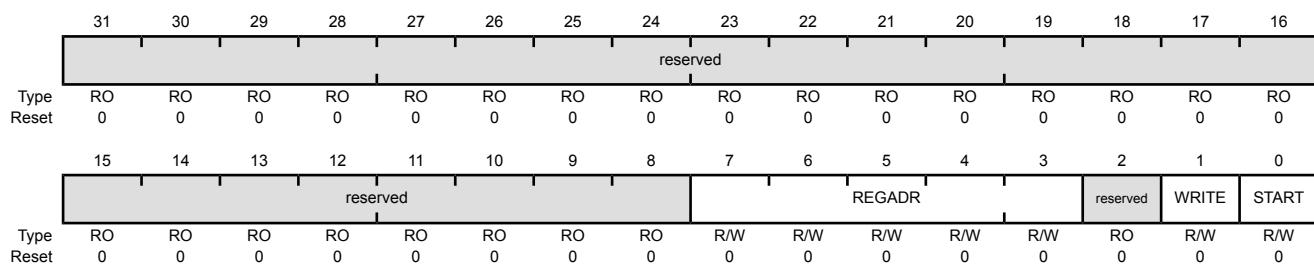
This register enables software to control the transfer of data to and from the MII Management registers in the Ethernet PHY layer. The address, name, type, reset configuration, and functional description of each of these registers can be found in Table 18-4 on page 865 and in “MII Management Register Descriptions” on page 893.

In order to initiate a *read* transaction from the MII Management registers, the **WRITE** bit must be cleared during the same cycle that the **START** bit is set.

In order to initiate a *write* transaction to the MII Management registers, the **WRITE** bit must be set during the same cycle that the **START** bit is set.

Ethernet MAC Management Control (MACMCTL)

Base 0x4004.8000
Offset 0x020
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	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.
7:3	REGADR	R/W	0x0	MII Register Address The REGADR bit field represents the MII Management register address for the next MII management interface transaction. Refer to Table 18-4 on page 865 for the PHY register offsets. Note that any address that is not valid in the register map should not be written to, and any data read should be ignored.
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	WRITE	R/W	0	MII Register Transaction Type Value Description 1 The next operation of the next MII management interface is a write transaction. 0 The next operation of the next MII management interface is a read transaction.

Bit/Field	Name	Type	Reset	Description
0	START	R/W	0	MII Register Transaction Enable
Value Description				
		1		The MII register located at REGADR is read (WRITE=0) or written (WRITE=1).
		0		No effect.

Register 10: Ethernet MAC Management Divider (MACMDV), offset 0x024

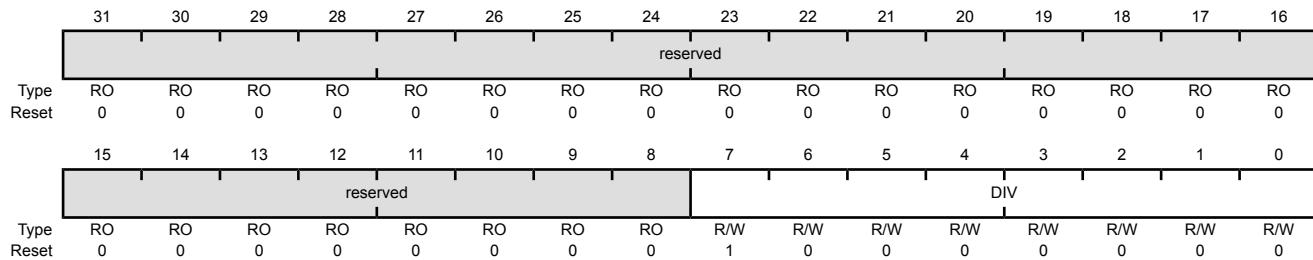
This register enables software to set the clock divider for the Management Data Clock (MDC). This clock is used to synchronize read and write transactions between the system and the MII Management registers. The frequency of the MDC clock can be calculated from the following formula:

$$F_{mdc} = \frac{F_{ipclk}}{2 \times (\text{MACMDV} + 1)}$$

The clock divider must be written with a value that ensures that the MDC clock does not exceed a frequency of 2.5 MHz.

Ethernet MAC Management Divider (MACMDV)

Base 0x4004.8000
Offset 0x024
Type R/W, reset 0x0000.0080



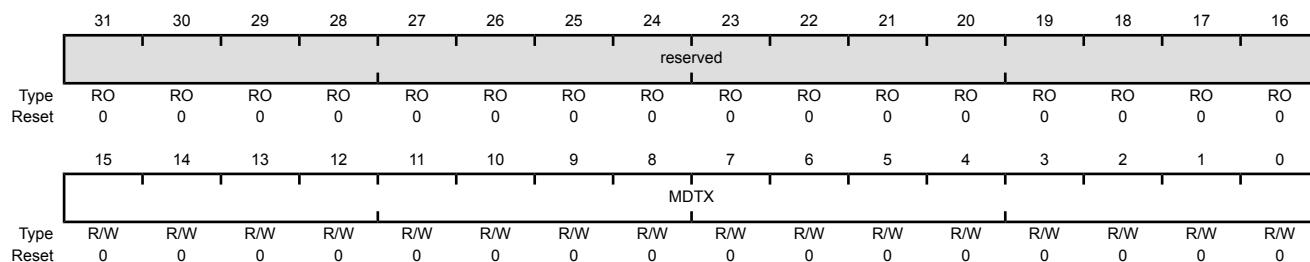
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIV	R/W	0x80	Clock Divider The DIV bits are used to set the clock divider for the MDC clock used to transmit data between the MAC and PHY layers.

Register 11: Ethernet MAC Management Transmit Data (MACMTXD), offset 0x02C

This register holds the next value to be written to the MII Management registers.

Ethernet MAC Management Transmit Data (MACMTXD)

Base 0x4004.8000
Offset 0x02C
Type R/W, reset 0x0000.0000



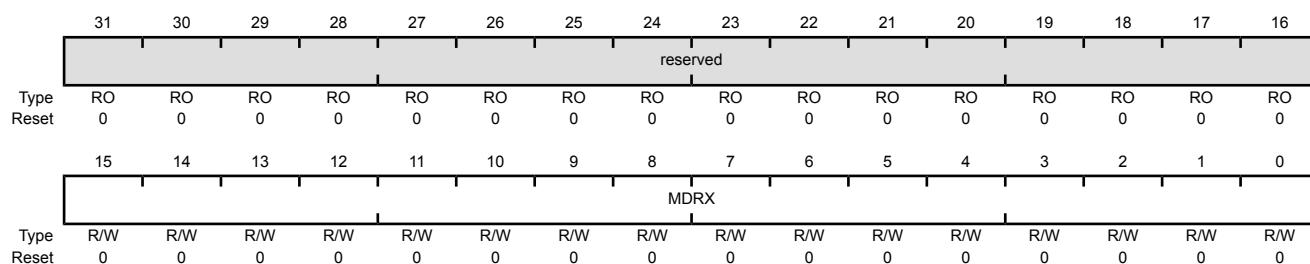
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MDTX	R/W	0x0000	MII Register Transmit Data The MDTX bits represent the data to be written in the next MII management transaction.

Register 12: Ethernet MAC Management Receive Data (MACMRXD), offset 0x030

This register holds the last value read from the MII Management registers.

Ethernet MAC Management Receive Data (MACMRXD)

Base 0x4004.8000
Offset 0x030
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	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	MDRX	R/W	0x0000	MII Register Receive Data The MDRX bits represent the data that was read in the previous MII management transaction.

Register 13: Ethernet MAC Number of Packets (MACNP), offset 0x034

This register holds the number of frames that are currently in the RX FIFO. When **NPR** is 0, there are no frames in the RX FIFO, and the **RXINT** bit is clear. When **NPR** is any other value, at least one frame is in the RX FIFO, and the **RXINT** bit in the **MACRIS** register is set.

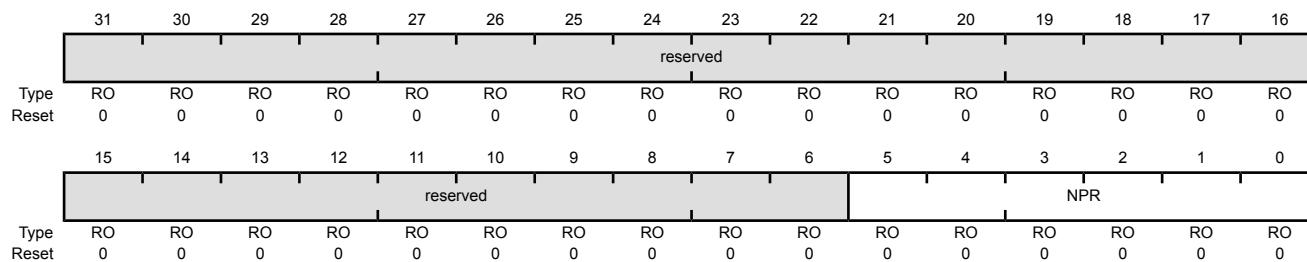
Note: The FCS bytes are not included in the **NPR** value. As a result, the **NPR** value could be zero before the FCS bytes are read from the FIFO. In addition, a new packet could be received before the **NPR** value reaches zero. To ensure that the entire packet is received, either use the DriverLib **EthernetPacketGet()** API or compare the number of bytes received to the **Length** field from the frame to determine when the packet has been completely read.

Ethernet MAC Number of Packets (MACNP)

Base 0x4004.8000

Offset 0x034

Type RO, reset 0x0000.0000



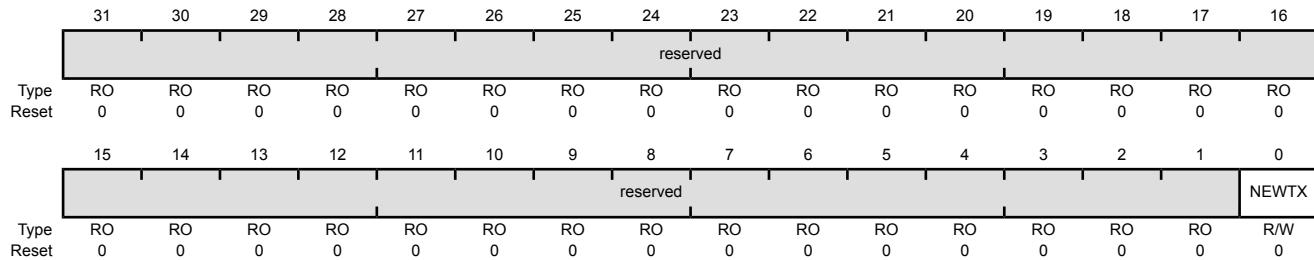
Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	NPR	RO	0x00	Number of Packets in Receive FIFO The NPR bits represent the number of packets stored in the RX FIFO. While the NPR field is greater than 0, the RXINT interrupt in the MACRIS register is set.

Register 14: Ethernet MAC Transmission Request (MACTR), offset 0x038

This register enables software to initiate the transmission of the frame currently located in the TX FIFO. Once the frame has been transmitted from the TX FIFO or a transmission error has been encountered, the NEWTX bit is automatically cleared.

Ethernet MAC Transmission Request (MACTR)

Base 0x4004.8000
Offset 0x038
Type R/W, reset 0x0000.0000



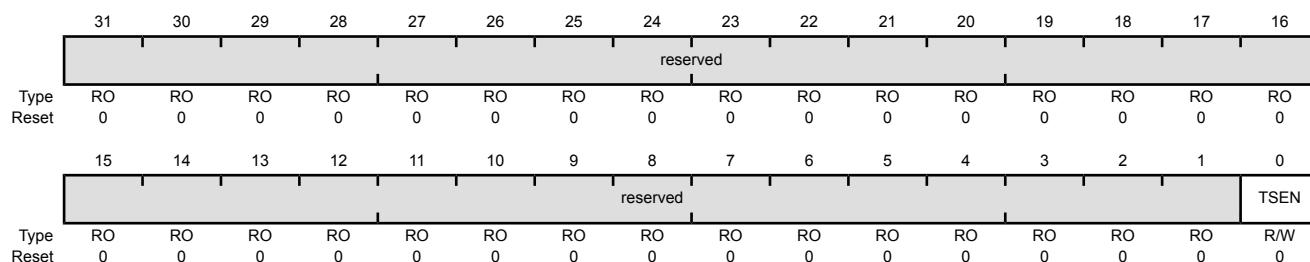
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	NEWTX	R/W	0	New Transmission
	Value	Description		
	1	Initiates an Ethernet transmission once the packet has been placed in the TX FIFO.		
	0	The transmission has completed.		
				If early transmission is being used (see the MACTHR register), this bit does not need to be set.

Register 15: Ethernet MAC Timer Support (MACTS), offset 0x03C

This register enables software to enable highly precise timing on the transmission and reception of frames. To enable this function, set the TSEN bit.

Ethernet MAC Timer Support (MACTS)

Base 0x4004.8000
Offset 0x03C
Type R/W, reset 0x0000.0000



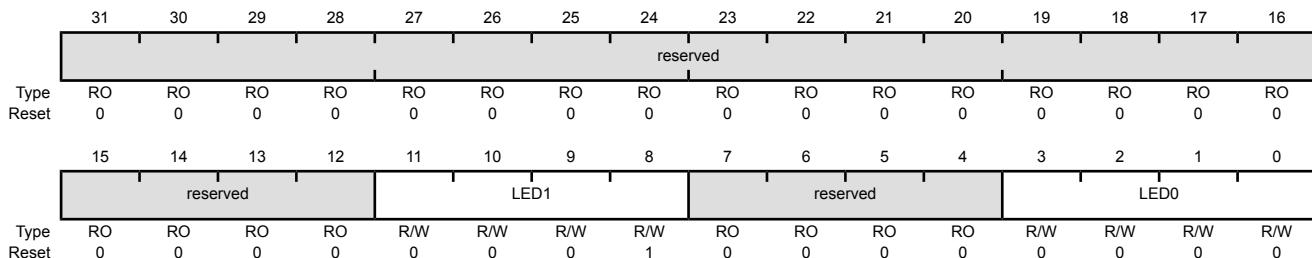
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	TSEN	R/W	0	Time Stamp Enable Value Description 1 The TX and RX interrupts are routed to the CCP inputs of General-Purpose Timer 3. 0 No effect.

Register 16: Ethernet MAC LED Encoding (MACLED), offset 0x040

This register enables software to select the source that causes the `LED1` and `LED0` signal to toggle.

Ethernet MAC LED Encoding (MACLED)

Base 0x4004.8000
Offset 0x040
Type R/W, reset 0x0000.0100



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.																		
11:8	LED1	R/W	0x1	<p>LED1 Source</p> <p>The <code>LED1</code> field selects the source that toggles the <code>LED1</code> signal.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Link OK</td> </tr> <tr> <td>0x1</td> <td>RX or TX Activity (Default LED1) Note that when RX or TX activity stops, the LED output is extended by 128 ms.</td> </tr> <tr> <td>0x2-0x4</td> <td>Reserved</td> </tr> <tr> <td>0x5</td> <td>100BASE-TX mode</td> </tr> <tr> <td>0x6</td> <td>10BASE-T mode</td> </tr> <tr> <td>0x7</td> <td>Full-Duplex</td> </tr> <tr> <td>0x8</td> <td>Link OK & Blink=RX or TX Activity</td> </tr> <tr> <td>0x9-0xF</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Description	0x0	Link OK	0x1	RX or TX Activity (Default LED1) Note that when RX or TX activity stops, the LED output is extended by 128 ms.	0x2-0x4	Reserved	0x5	100BASE-TX mode	0x6	10BASE-T mode	0x7	Full-Duplex	0x8	Link OK & Blink=RX or TX Activity	0x9-0xF	Reserved
Value	Description																					
0x0	Link OK																					
0x1	RX or TX Activity (Default LED1) Note that when RX or TX activity stops, the LED output is extended by 128 ms.																					
0x2-0x4	Reserved																					
0x5	100BASE-TX mode																					
0x6	10BASE-T mode																					
0x7	Full-Duplex																					
0x8	Link OK & Blink=RX or TX Activity																					
0x9-0xF	Reserved																					
7:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																		

Bit/Field	Name	Type	Reset	Description																		
3:0	LED0	R/W	0x0	<p>LED0 Source The LED0 field selects the source that toggles the LED0 signal.</p> <table><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0x0</td><td>Link OK (Default LED0)</td></tr><tr><td>0x1</td><td>RX or TX Activity Note that when RX or TX activity stops, the LED output is extended by 128 ms.</td></tr><tr><td>0x2-0x4</td><td>Reserved</td></tr><tr><td>0x5</td><td>100BASE-TX mode</td></tr><tr><td>0x6</td><td>10BASE-T mode</td></tr><tr><td>0x7</td><td>Full-Duplex</td></tr><tr><td>0x8</td><td>Link OK & Blink=RX or TX Activity</td></tr><tr><td>0x9-0xF</td><td>Reserved</td></tr></tbody></table>	Value	Description	0x0	Link OK (Default LED0)	0x1	RX or TX Activity Note that when RX or TX activity stops, the LED output is extended by 128 ms.	0x2-0x4	Reserved	0x5	100BASE-TX mode	0x6	10BASE-T mode	0x7	Full-Duplex	0x8	Link OK & Blink=RX or TX Activity	0x9-0xF	Reserved
Value	Description																					
0x0	Link OK (Default LED0)																					
0x1	RX or TX Activity Note that when RX or TX activity stops, the LED output is extended by 128 ms.																					
0x2-0x4	Reserved																					
0x5	100BASE-TX mode																					
0x6	10BASE-T mode																					
0x7	Full-Duplex																					
0x8	Link OK & Blink=RX or TX Activity																					
0x9-0xF	Reserved																					

Register 17: Ethernet PHY MDIX (MDIX), offset 0x044

This register enables the transmit and receive lines to be reversed in order to implement the MDI/MDI-X functionality. Software can implement the MDI/MDI-X configuration by using any available timer resource such as SysTick (see “System Timer (SysTick)” on page 116 for more information) to implement this functionality. Once the Ethernet Controller has been configured and enabled, software should check to see if the `LINK` bit in the **MR1** register has been set within approximately 1 s; if not, set the `EN` bit of the **MDIX** register to switch the reverse the transmit and receive lines to the PHY layer. Software should check the `LINK` bit again after approximately another 1 s and if no link has been established, the `EN` bit should be cleared. Software must continue to change the termination back and forth by setting and clearing the `EN` bit every 1 s until a link is established.

Ethernet PHY MDIX (MDIX)

Base 0x4004.8000
Offset 0x044
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO	R/W														
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	EN	R/W	0	MDI/MDI-X Enable
		Value	Description	
		1	The transmit and receive signals are switched such that data is received on the transmit signals <code>TXOP</code> and <code>TXON</code> ; data is transmitted on the receive signals <code>RXIP</code> and <code>RXIN</code>	
		0	No effect.	

18.7 MII Management Register Descriptions

The *IEEE 802.3 standard* specifies a register set for controlling and gathering status from the PHY layer. The registers are collectively known as the MII Management registers. The **Ethernet MAC Management Control (MACMCTL)** register is used to access the MII Management registers, see page 883. All addresses given are absolute. Addresses not listed are reserved; these addresses should not be written to and any data read should be ignored. Also see “Ethernet MAC Register Descriptions” on page 867.

Register 18: Ethernet PHY Management Register 0 – Control (MR0), address 0x00

This register enables software to configure the operation of the PHY layer. The default settings of these registers are designed to initialize the Ethernet Controller to a normal operational mode without configuration.

Ethernet PHY Management Register 0 – Control (MR0)

Base 0x4004.8000

Address 0x00

Type R/W, reset 0x1000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RESET	LOOPBK	SPEEDSL	ANEGEN	PWRDN	ISO	RANEG	DUPLEX	COLT				reserved			
Reset	R/W 0	R/W 0	R/W 0	R/W 1	R/W 0											

Bit/Field	Name	Type	Reset	Description
15	RESET	R/W	0	Reset Registers
		Value	Description	
		1	The PHY layer registers reset to their default state and the internal state machines are reinitialized.	
		0	No effect.	
		Once the reset operation has completed, this bit is automatically cleared by hardware.		
14	LOOPBK	R/W	0	Loopback Mode
		Value	Description	
		1	Enables the Loopback mode of operation. The receiver ignores external inputs and receives the data that is transmitted by the transmitter.	
		0	No effect.	
13	SPEEDSL	R/W	0	Speed Select
		Value	Description	
		1	Enables the 100 Mbps mode of operation (100BASE-TX).	
		0	Enables the 10 Mbps mode of operation (10BASE-T).	
12	ANEGEN	R/W	1	Auto-Negotiation Enable
		Value	Description	
		1	Enables the auto-negotiation process.	
		0	No effect.	

Bit/Field	Name	Type	Reset	Description						
11	PWRDN	R/W	0	<p>Power Down</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The PHY layer is configured to be in a low-power consuming state. All data on the data inputs is ignored.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table>	Value	Description	1	The PHY layer is configured to be in a low-power consuming state. All data on the data inputs is ignored.	0	No effect.
Value	Description									
1	The PHY layer is configured to be in a low-power consuming state. All data on the data inputs is ignored.									
0	No effect.									
10	ISO	R/W	0	<p>Isolate</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The transmit and receive data paths are isolated and all data being transmitted and received is ignored.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table>	Value	Description	1	The transmit and receive data paths are isolated and all data being transmitted and received is ignored.	0	No effect.
Value	Description									
1	The transmit and receive data paths are isolated and all data being transmitted and received is ignored.									
0	No effect.									
9	RANEG	R/W	0	<p>Restart Auto-Negotiation</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Restarts the auto-negotiation process.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table> <p>Once the restart has initiated, this bit is automatically cleared by hardware.</p>	Value	Description	1	Restarts the auto-negotiation process.	0	No effect.
Value	Description									
1	Restarts the auto-negotiation process.									
0	No effect.									
8	DUPLEX	R/W	0	<p>Set Duplex Mode</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Enables the Full-Duplex mode of operation. This bit can be set by software in a manual configuration process or by the auto-negotiation process.</td></tr> <tr> <td>0</td><td>Enables the Half-Duplex mode of operation. Note that in 10BASE-T half-duplex mode, the transmitted data is looped back on the receive path.</td></tr> </tbody> </table>	Value	Description	1	Enables the Full-Duplex mode of operation. This bit can be set by software in a manual configuration process or by the auto-negotiation process.	0	Enables the Half-Duplex mode of operation. Note that in 10BASE-T half-duplex mode, the transmitted data is looped back on the receive path.
Value	Description									
1	Enables the Full-Duplex mode of operation. This bit can be set by software in a manual configuration process or by the auto-negotiation process.									
0	Enables the Half-Duplex mode of operation. Note that in 10BASE-T half-duplex mode, the transmitted data is looped back on the receive path.									
7	COLT	R/W	0	<p>Collision Test</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Enables the Collision Test mode of operation.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table> <p>The COLT bit is set after the initiation of a transmission and is cleared once the transmission is halted.</p>	Value	Description	1	Enables the Collision Test mode of operation.	0	No effect.
Value	Description									
1	Enables the Collision Test mode of operation.									
0	No effect.									
6:0	reserved	R/W	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. These bits should always be written as zero.						

Register 19: Ethernet PHY Management Register 1 – Status (MR1), address 0x01

This register enables software to determine the capabilities of the PHY layer and perform its initialization and operation appropriately.

Ethernet PHY Management Register 1 – Status (MR1)

Base 0x4004.8000

Address 0x01

Type RO, reset 0x7809

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	reserved	100X_F	100X_H	10T_F	10T_H		reserved			ANEGC	RFAULT	ANEGA	LINK	JAB	EXTD	
Reset	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RC	RO	RO	RC	RO	

Bit/Field	Name	Type	Reset	Description
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	100X_F	RO	1	100BASE-TX Full-Duplex Mode
		Value	Description	
		1	The Ethernet Controller is capable of supporting 100BASE-TX Full-Duplex mode.	
		0	The Ethernet Controller is not capable of supporting 100BASE-TX Full-Duplex mode.	
13	100X_H	RO	1	100BASE-TX Half-Duplex Mode
		Value	Description	
		1	The Ethernet Controller is capable of supporting 100BASE-TX Half-Duplex mode.	
		0	The Ethernet Controller is not capable of supporting 100BASE-TX Half-Duplex mode.	
12	10T_F	RO	1	10BASE-T Full-Duplex Mode
		Value	Description	
		1	The Ethernet Controller is capable of supporting 10BASE-T Full-Duplex mode.	
		0	The Ethernet Controller is not capable of supporting 10BASE-T Full-Duplex mode.	
11	10T_H	RO	1	10BASE-T Half-Duplex Mode
		Value	Description	
		1	The Ethernet Controller is capable of supporting 10BASE-T Half-Duplex mode.	
		0	The Ethernet Controller is not capable of supporting 10BASE-T Half-Duplex mode.	

Bit/Field	Name	Type	Reset	Description
10:6	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	ANEGC	RO	0	Auto-Negotiation Complete Value Description 1 The auto-negotiation process has been completed and that the extended registers defined by the auto-negotiation protocol are valid. 0 The auto-negotiation process is not complete.
4	RFAULT	RC	0	Remote Fault Value Description 1 A remote fault condition has been detected. 0 A remote fault condition has not been detected. This bit remains set until it is read, even if the condition no longer exists.
3	ANEGA	RO	1	Auto-Negotiation Value Description 1 The Ethernet Controller has the ability to perform auto-negotiation. 0 The Ethernet Controller does not have the ability to perform auto-negotiation.
2	LINK	RO	0	Link Made Value Description 1 A valid link has been established by the Ethernet Controller. 0 A valid link has not been established by the Ethernet Controller.
1	JAB	RC	0	Jabber Condition Value Description 1 A jabber condition has been detected by the Ethernet Controller. 0 A jabber condition has not been detected by the Ethernet Controller. This bit remains set until it is read, even if the jabber condition no longer exists.
0	EXTD	RO	1	Extended Capabilities Value Description 1 The Ethernet Controller provides an extended set of capabilities that can be accessed through the extended register set. 0 The Ethernet Controller does not provide extended capabilities.

Register 20: Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2), address 0x02

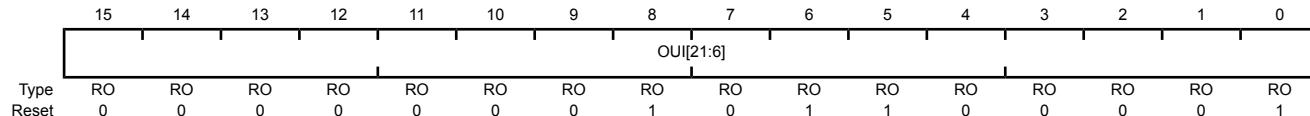
This register, along with **MR3**, provides a 32-bit value indicating the manufacturer, model, and revision information.

Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2)

Base 0x4004.8000

Address 0x02

Type RO, reset 0x0161



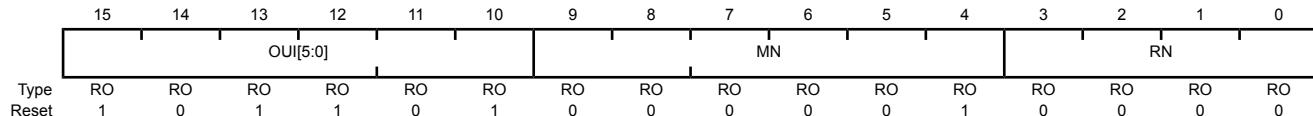
Bit/Field	Name	Type	Reset	Description
15:0	OUI[21:6]	RO	0x0161	Organizational Unique Identifier[21:6] This field, along with the OUI[5:0] field in MR3 , makes up the Organizational Unique Identifier indicating the PHY manufacturer.

Register 21: Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3), address 0x03

This register, along with **MR2**, provides a 32-bit value indicating the manufacturer, model, and revision information.

Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3)

Base 0x4004.8000
Address 0x03
Type RO, reset 0xB410



Bit/Field	Name	Type	Reset	Description
15:10	OUI[5:0]	RO	0x2D	Organizationally Unique Identifier[5:0] This field, along with the OUI [21 : 6] field in MR2 , makes up the Organizationally Unique Identifier indicating the PHY manufacturer.
9:4	MN	RO	0x01	Model Number The MN field represents the Model Number of the PHY.
3:0	RN	RO	0x0	Revision Number The RN field represents the Revision Number of the PHY implementation.

Register 22: Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4), address 0x04

This register provides the advertised abilities of the Ethernet Controller used during auto-negotiation. Bits 8:5 represent the Technology Ability Field bits. This field can be overwritten by software to auto-negotiate to an alternate common technology. Writing to this register has no effect until auto-negotiation is re-initiated by setting the RANEG bit in the **MR0** register.

Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4)

Base 0x4004.8000

Address 0x04

Type R/W, reset 0x01E1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	NP	reserved	RF	reserved	reserved	reserved	A3	A2	A1	A0			S			
Reset	RO 0	RO 0	R/W 0	RO 0	RO 0	RO 0	R/W 1	R/W 1	R/W 1	R/W 1	RO 0	RO 0	RO 0	RO 0	RO 1	

Bit/Field	Name	Type	Reset	Description
15	NP	RO	0	Next Page
				Value Description
			1	The Ethernet Controller is capable of Next Page exchanges to provide more detailed information on the PHY layer's capabilities.
			0	The Ethernet Controller is not capable of Next Page exchanges.
14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	RF	R/W	0	Remote Fault
				Value Description
			1	Indicates to the link partner that a Remote Fault condition has been encountered.
			0	No Remote Fault condition has been encountered.
12:9	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	A3	R/W	1	Technology Ability Field [3]
				Value Description
			1	The Ethernet Controller supports the 100Base-TX full-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the MR0 register.
			0	The Ethernet Controller does not support the 100Base-TX full-duplex signaling protocol.

Bit/Field	Name	Type	Reset	Description						
7	A2	R/W	1	<p>Technology Ability Field [2]</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The Ethernet Controller supports the 100Base-TX half-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the MRO register.</td></tr> <tr> <td>0</td><td>The Ethernet Controller does not support the 100Base-TX half-duplex signaling protocol.</td></tr> </tbody> </table>	Value	Description	1	The Ethernet Controller supports the 100Base-TX half-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the MRO register.	0	The Ethernet Controller does not support the 100Base-TX half-duplex signaling protocol.
Value	Description									
1	The Ethernet Controller supports the 100Base-TX half-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the MRO register.									
0	The Ethernet Controller does not support the 100Base-TX half-duplex signaling protocol.									
6	A1	R/W	1	<p>Technology Ability Field [1]</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The Ethernet Controller supports the 10BASE-T full-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the MRO register.</td></tr> <tr> <td>0</td><td>The Ethernet Controller does not support the 10BASE-T full-duplex signaling protocol.</td></tr> </tbody> </table>	Value	Description	1	The Ethernet Controller supports the 10BASE-T full-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the MRO register.	0	The Ethernet Controller does not support the 10BASE-T full-duplex signaling protocol.
Value	Description									
1	The Ethernet Controller supports the 10BASE-T full-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the MRO register.									
0	The Ethernet Controller does not support the 10BASE-T full-duplex signaling protocol.									
5	A0	R/W	1	<p>Technology Ability Field [0]</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The Ethernet Controller supports the 10BASE-T half-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the MRO register.</td></tr> <tr> <td>0</td><td>The Ethernet Controller does not support the 10BASE-T half-duplex signaling protocol.</td></tr> </tbody> </table>	Value	Description	1	The Ethernet Controller supports the 10BASE-T half-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the MRO register.	0	The Ethernet Controller does not support the 10BASE-T half-duplex signaling protocol.
Value	Description									
1	The Ethernet Controller supports the 10BASE-T half-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be cleared and auto-negotiation re-initiated with the RANEG bit in the MRO register.									
0	The Ethernet Controller does not support the 10BASE-T half-duplex signaling protocol.									
4:0	S	RO	0x1	<p>Selector Field</p> <p>This field encodes 32 possible messages for communicating between Ethernet Controllers. This field is hard-coded to 0x01, indicating that the Stellaris Ethernet Controller is <i>IEEE 802.3</i> compliant.</p>						

Register 23: Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5), address 0x05

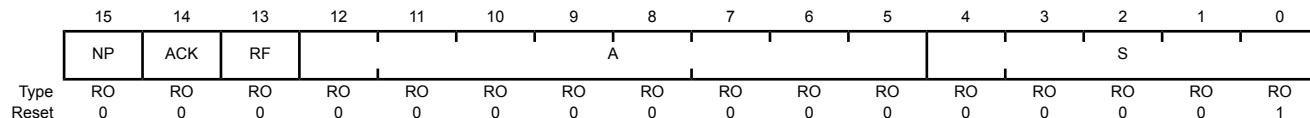
This register provides the advertised abilities of the link partner's Ethernet Controller that are received and stored during auto-negotiation.

Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5)

Base 0x4004.8000

Address 0x05

Type RO, reset 0x0001



Bit/Field	Name	Type	Reset	Description
15	NP	RO	0	<p>Value Description</p> <p>1 The link partner's Ethernet Controller is capable of Next page exchanges to provide more detailed information on the Ethernet Controller's capabilities.</p> <p>0 The link partner's Ethernet Controller is not capable of Next page exchanges</p>
14	ACK	RO	0	<p>Acknowledge</p> <p>Value Description</p> <p>1 The Ethernet Controller has successfully received the link partner's advertised abilities during auto-negotiation.</p> <p>0 The Ethernet Controller has not received the link partner's advertised abilities during auto-negotiation.</p>
13	RF	RO	0	<p>Remote Fault</p> <p>Value Description</p> <p>1 The link partner is indicating that a Remote Fault condition has been encountered.</p> <p>0 The link partner is not indicating that a Remote Fault condition has been encountered.</p>
12:5	A	RO	0x00	<p>Technology Ability Field</p> <p>This field encodes individual technologies that are supported by the Ethernet Controller. See the MR4 register for definitions. Note that bits [12:9] describe functions that are not implemented on the Stellaris Ethernet Controller. Refer to the IEEE 802.3 standard for definitions.</p>

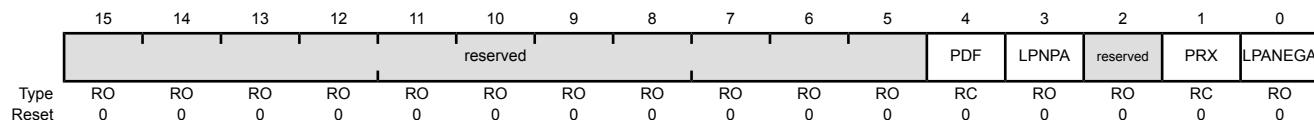
Bit/Field	Name	Type	Reset	Description
4:0	S	RO	0x01	Selector Field This field encodes possible messages for communicating between Ethernet Controllers.
				Value Description
			0x00	Reserved
			0x01	IEEE Std 802.3
			0x02	IEEE Std 802.9 ISLAN-16T
			0x03	IEEE Std 802.5
			0x04	IEEE Std 1394
			0x05–0x1F	Reserved

Register 24: Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6), address 0x06

This register enables software to determine the auto-negotiation and next page capabilities of the Ethernet Controller and the link partner after auto-negotiation.

Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6)

Base 0x4004.8000
Address 0x06
Type RO, reset 0x0000



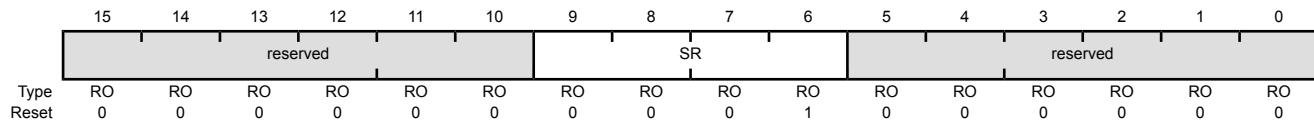
Bit/Field	Name	Type	Reset	Description
15:5	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	PDF	RC	0	<p>Parallel Detection Fault</p> <p>Value Description</p> <p>1 More than one technology was detected at link up.</p> <p>0 Only one technology was detected at link up.</p> <p>This bit is automatically cleared when read.</p>
3	LPNPA	RO	0	<p>Link Partner is Next Page Able</p> <p>Value Description</p> <p>1 The link partner is enabled to support next page.</p> <p>0 The link partner is not enabled to support next page.</p>
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	PRX	RC	0	<p>New Page Received</p> <p>Value Description</p> <p>1 A new page has been received from the link partner and stored.</p> <p>0 A new page has not been received.</p> <p>This bit is automatically cleared when read.</p>
0	LPANEGA	RO	0	<p>Link Partner is Auto-Negotiation Able</p> <p>Value Description</p> <p>1 The link partner is enabled to support auto-negotiation.</p> <p>0 The link partner is not enabled to support auto-negotiation.</p>

Register 25: Ethernet PHY Management Register 16 – Vendor-Specific (MR16), address 0x10

This register contains a silicon revision identifier.

Ethernet PHY Management Register 16 – Vendor-Specific (MR16)

Base 0x4004.8000
Address 0x10
Type RO, reset 0x0040



Bit/Field	Name	Type	Reset	Description
15:10	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.
9:6	SR	RO	0x1	Silicon Revision Identifier This field contains the four-bit identifier for the silicon revision.
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 26: Ethernet PHY Management Register 17 – Mode Control/Status (MR17), address 0x11

This register provides the means for controlling and observing various PHY layer modes.

Ethernet PHY Management Register 17 – Mode Control/Status (MR17)

Base 0x4004.8000
Address 0x11
Type R/W, reset 0x0002

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	reserved	FASTRIP	EDPD	reserved	LSQE	reserved	FASTEAST			reserved				FGLS	ENON	reserved
Reset	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	RO 0	RO 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	R/W 0	RO 1	R/W 0

Bit/Field	Name	Type	Reset	Description
15	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. Important: This bit must always be written with a 0 to ensure proper operation.
14	FASTRIP	R/W	0	10-BASE-T Fast Mode Enable Value Description 1 Enables PHYT_10 test mode. 0 No effect.
13	EDPD	R/W	0	Enable Energy Detect Power Down Value Description 1 Enables the Energy Detect Power Down mode. 0 No effect.
12	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. Important: This bit must always be written with a 0 to ensure proper operation.
11	LSQE	R/W	0	Low Squelch Enable Value Description 1 Enables a lower threshold meaning more sensitivity to the signal levels. 0 No effect.
10:9	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	FAATEST	R/W	0	<p>Auto-Negotiation Test Mode</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Enables the Auto-Negotiation Test mode.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table>	Value	Description	1	Enables the Auto-Negotiation Test mode.	0	No effect.
Value	Description									
1	Enables the Auto-Negotiation Test mode.									
0	No effect.									
7:3	reserved	R/W	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> <p>Important: This bit must always be written with a 0 to ensure proper operation.</p>						
2	FGLS	R/W	0	<p>Force Good Link Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Forces the 100BASE-T link to be active.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table> <p>Note: This bit should only be set when testing.</p>	Value	Description	1	Forces the 100BASE-T link to be active.	0	No effect.
Value	Description									
1	Forces the 100BASE-T link to be active.									
0	No effect.									
1	ENON	RO	1	<p>Energy On</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Energy is detected on the line.</td></tr> <tr> <td>0</td><td>Valid energy has not been detected on the line within 256 ms.</td></tr> </tbody> </table> <p>This bit is set by a hardware reset, but is unaffected by a software reset.</p>	Value	Description	1	Energy is detected on the line.	0	Valid energy has not been detected on the line within 256 ms.
Value	Description									
1	Energy is detected on the line.									
0	Valid energy has not been detected on the line within 256 ms.									
0	reserved	R/W	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> <p>Important: This bit must always be written with a 0 to ensure proper operation.</p>						

Register 27: Ethernet PHY Management Register 27 – Special Control/Status (MR27), address 0x1B

This register shows the status of the 10BASE-T polarity.

Ethernet PHY Management Register 27 – Special Control/Status (MR27)

Base 0x4004.8000

Address 0x1B

Type RO, reset -

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											XPOL	reserved			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
15:5	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	XPOL	RO	0	Polarity State of 10 BASE-T
		Value	Description	
		1	The 10BASE-T is reversed polarity.	
		0	The 10BASE-T is normal polarity.	
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 28: Ethernet PHY Management Register 29 – Interrupt Status (MR29), address 0x1D

This register contains information about the source of PHY layer interrupts. Reading this register clears any bits that are set. The **PHYINT** bit is set in the **MACRIS/MACIACK** register whenever any of the bits in this register are set.

Ethernet PHY Management Register 29 – Interrupt Status (MR29)

Base 0x4004.8000
Address 0x1D
Type RO, reset 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO	EONIS	ANCOMPIS	RFLTIS	LDIS	LPACKIS	PDFIS	PRXIS	reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description
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	EONIS	RO	0	ENERGYON Interrupt Value Description 1 An interrupt has been generated due to the ENON bit being set in the MR17 register. 0 No interrupt. This bit is cleared by reading the value.
6	ANCOMPIS	RO	0	Auto-Negotiation Complete Interrupt Value Description 1 An interrupt has been generated due to the completion of auto negotiation. 0 No interrupt. This bit is cleared by reading the value.
5	RFLTIS	RO	0	Remote Fault Interrupt Value Description 1 An interrupt has been generated due to the detection of a Remote Fault. 0 No interrupt. This bit is cleared by reading the value.
4	LDIS	RO	0	Link Down Interrupt Value Description 1 An interrupt has been generated because the LINK bit in MR1 is clear. 0 No interrupt. This bit is cleared by reading the value.

Bit/Field	Name	Type	Reset	Description						
3	LPACKIS	RO	0	<p>Auto-Negotiation LP Acknowledge</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An interrupt has been generated due to the reception of an acknowledge message from the link partner during auto-negotiation.</td></tr> <tr> <td>0</td><td>No interrupt.</td></tr> </tbody> </table> <p>This bit is cleared by reading the value.</p>	Value	Description	1	An interrupt has been generated due to the reception of an acknowledge message from the link partner during auto-negotiation.	0	No interrupt.
Value	Description									
1	An interrupt has been generated due to the reception of an acknowledge message from the link partner during auto-negotiation.									
0	No interrupt.									
2	PDFIS	RO	0	<p>Parallel Detection Fault</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An interrupt has been generated due to the detection of a parallel detection fault during auto negotiation.</td></tr> <tr> <td>0</td><td>No interrupt.</td></tr> </tbody> </table> <p>This bit is cleared by reading the value.</p>	Value	Description	1	An interrupt has been generated due to the detection of a parallel detection fault during auto negotiation.	0	No interrupt.
Value	Description									
1	An interrupt has been generated due to the detection of a parallel detection fault during auto negotiation.									
0	No interrupt.									
1	PRXIS	RO	0	<p>Auto Negotiation Page Received</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An interrupt has been generated due to the reception of an auto negotiation page from the link partner.</td></tr> <tr> <td>0</td><td>No interrupt.</td></tr> </tbody> </table> <p>This bit is cleared by reading the value.</p>	Value	Description	1	An interrupt has been generated due to the reception of an auto negotiation page from the link partner.	0	No interrupt.
Value	Description									
1	An interrupt has been generated due to the reception of an auto negotiation page from the link partner.									
0	No 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 29: Ethernet PHY Management Register 30 – Interrupt Mask (MR30), address 0x1E

This register enables interrupts to be generated by the various sources of PHY layer interrupts.

Ethernet PHY Management Register 30 – Interrupt Mask (MR30)

Base 0x4004.8000
Address 0x1E
Type R/W, reset 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO	R/W	EONIM	ANCOMPIM	RFLTIM	LDIM	LPACKIM	PDFIM	PRXIM	reserved						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
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	EONIM	R/W	0	ENERGYON Interrupt Enabled
		Value	Description	
		1	An interrupt is sent to the interrupt controller when the EONIS bit in the MR29 register is set.	
		0	The EONIS interrupt is suppressed and not sent to the interrupt controller.	
6	ANCOMPIM	R/W	0	Auto-Negotiation Complete Interrupt Enabled
		Value	Description	
		1	An interrupt is sent to the interrupt controller when the ANCOMPIS bit in the MR29 register is set.	
		0	The ANCOMPIS interrupt is suppressed and not sent to the interrupt controller.	
5	RFLTIM	R/W	0	Remote Fault Interrupt Enabled
		Value	Description	
		1	An interrupt is sent to the interrupt controller when the RFLTIS bit in the MR29 register is set.	
		0	The RFLTIS interrupt is suppressed and not sent to the interrupt controller.	
4	LDIM	R/W	0	Link Down Interrupt Enabled
		Value	Description	
		1	An interrupt is sent to the interrupt controller when the LDIS bit in the MR29 register is set.	
		0	The LDIS interrupt is suppressed and not sent to the interrupt controller.	

Bit/Field	Name	Type	Reset	Description						
3	LPACKIM	R/W	0	<p>Auto-Negotiation LP Acknowledge Enabled</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An interrupt is sent to the interrupt controller when the LPACKIS bit in the MR29 register is set.</td></tr> <tr> <td>0</td><td>The LPACKIS interrupt is suppressed and not sent to the interrupt controller.</td></tr> </tbody> </table>	Value	Description	1	An interrupt is sent to the interrupt controller when the LPACKIS bit in the MR29 register is set.	0	The LPACKIS interrupt is suppressed and not sent to the interrupt controller.
Value	Description									
1	An interrupt is sent to the interrupt controller when the LPACKIS bit in the MR29 register is set.									
0	The LPACKIS interrupt is suppressed and not sent to the interrupt controller.									
2	PDFIM	R/W	0	<p>Parallel Detection Fault Enabled</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An interrupt is sent to the interrupt controller when the PDFIS bit in the MR29 register is set.</td></tr> <tr> <td>0</td><td>The PDFIS interrupt is suppressed and not sent to the interrupt controller.</td></tr> </tbody> </table>	Value	Description	1	An interrupt is sent to the interrupt controller when the PDFIS bit in the MR29 register is set.	0	The PDFIS interrupt is suppressed and not sent to the interrupt controller.
Value	Description									
1	An interrupt is sent to the interrupt controller when the PDFIS bit in the MR29 register is set.									
0	The PDFIS interrupt is suppressed and not sent to the interrupt controller.									
1	PRXIM	R/W	0	<p>Auto Negotiation Page Received Enabled</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An interrupt is sent to the interrupt controller when the PRXIS bit in the MR29 register is set.</td></tr> <tr> <td>0</td><td>The PRXIS interrupt is suppressed and not sent to the interrupt controller.</td></tr> </tbody> </table>	Value	Description	1	An interrupt is sent to the interrupt controller when the PRXIS bit in the MR29 register is set.	0	The PRXIS interrupt is suppressed and not sent to the interrupt controller.
Value	Description									
1	An interrupt is sent to the interrupt controller when the PRXIS bit in the MR29 register is set.									
0	The PRXIS interrupt is suppressed and not sent to the interrupt controller.									
0	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.						

Register 30: Ethernet PHY Management Register 31 – PHY Special Control/Status (MR31), address 0x1F

This register provides special control and status for the PHY layer.

Ethernet PHY Management Register 31 – PHY Special Control/Status (MR31)

Base 0x4004.8000
Address 0x1F
Type R/W, reset 0x0040

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	R/W	R/W	R/W	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
	reserved		AUTODONE		reserved							SPEED		reserved		SCRDIS

Bit/Field	Name	Type	Reset	Description
15:13	reserved	R/W	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.
				Important: This bit field must always be written with a 0 to ensure proper operation.
12	AUTODONE	RO	0	Auto Negotiation Done
				Value Description
			1	Auto negotiation is complete.
			0	Auto negotiation is not complete.
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:2	SPEED	RO	0x0	HCD Speed Value
				Value Description
			0x0	Reserved
			0x1	10BASE-T half duplex
			0x2	100BASE-T half duplex
			0x3-0x4	Reserved
			0x5	10BASE-T full duplex
			0x6	100BASE-T full duplex
			0x7	Reserved
1	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.
0	SCRDIS	R/W	0	Scramble Disable
				Value Description
			1	Disables data scrambling.
			0	Enables data scrambling.

19 Universal Serial Bus (USB) Controller

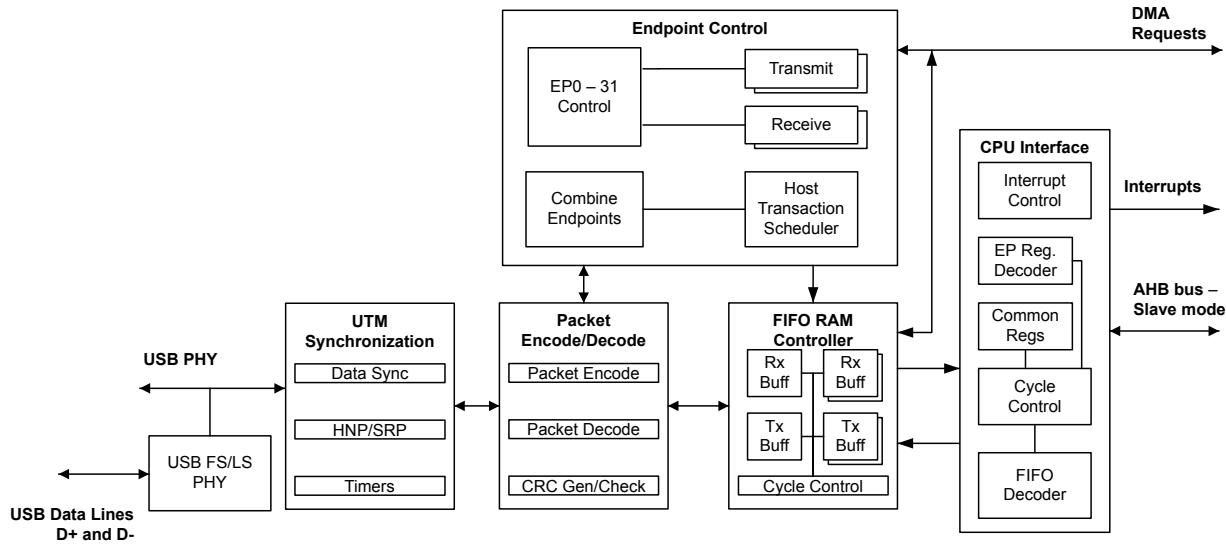
The Stellaris® USB controller operates as a full-speed or low-speed function controller during point-to-point communications with USB Host, Device, or OTG functions. The controller complies with the USB 2.0 standard, which includes SUSPEND and RESUME signaling. 32 endpoints including two hard-wired for control transfers (one endpoint for IN and one endpoint for OUT) plus 30 endpoints defined by firmware along with a dynamic sizable FIFO support multiple packet queueing. μDMA 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 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
- 32 endpoints
 - 1 dedicated control IN endpoint and 1 dedicated control OUT endpoint
 - 15 configurable IN endpoints and 15 configurable OUT endpoints
- 4 KB dedicated endpoint memory: one endpoint may be defined for double-buffered 1023-byte isochronous packet size
- VBUS droop and valid ID detection and interrupt
- Efficient transfers using Micro Direct Memory Access Controller (μDMA)
 - Separate channels for transmit and receive for up to three IN endpoints and three OUT endpoints
 - Channel requests asserted when FIFO contains required amount of data

19.1 Block Diagram

Figure 19-1. USB Module Block Diagram



19.2 Signal Description

Table 19-1 on page 915 and Table 19-2 on page 916 list the external signals of the USB controller and describe the function of each. Some USB controller signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these USB signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 447) should be set to choose the USB function. The number in parentheses is the encoding that must be programmed into the PMC_n field in the **GPIO Port Control (GPIOPCTL)** register (page 465) to assign the USB signal to the specified GPIO port pin. The **USB0VBUS** and **USB0ID** signals are configured by clearing the appropriate DEN bit in the **GPIO Digital Enable (GPIODEN)** register. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 423. The remaining signals (with the word "fixed" in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

Note: When used in OTG mode, **USB0VBUS** and **USB0ID** do not require any configuration as they are dedicated pins for the USB controller and directly connect to the USB connector's VBUS and ID signals. If the USB controller is used as either a dedicated Host or Device, the **DEVMODOTG** and **DEVMOD** bits in the **USB General-Purpose Control and Status (USBGPCS)** register can be used to connect the **USB0VBUS** and **USB0ID** inputs to fixed levels internally, freeing the PB0 and PB1 pins for GPIO use. For proper self-powered Device operation, the VBUS value must still be monitored to assure that if the Host removes VBUS, the self-powered Device disables the D+/D- pull-up resistors. This function can be accomplished by connecting a standard GPIO to VBUS.

Table 19-1. Signals for USB (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
USB0DM	70	fixed	I/O	Analog	Bidirectional differential data pin (D- per USB specification).

Table 19-1. Signals for USB (100LQFP) (continued)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
USB0DP	71	fixed	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	Optionally used in Host mode to control an external power source to supply power to the USB bus.
USB0ID	66	PB0	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 the A side of the cable and pulled up is the B side).
USB0PFLT	22 23 35 65 74 76 87	PC7 (6) PC6 (7) PA7 (8) PB3 (8) PE0 (9) PH4 (4) PJ1 (9)	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
USB0RBIAS	73	fixed	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.
USB0VBUS	67	PB1	I/O	Analog	This signal is used during the session request 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 has TTL-compatible voltage levels.

Table 19-2. Signals for USB (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
USB0DM	C11	fixed	I/O	Analog	Bidirectional differential data pin (D- per USB specification).
USB0DP	C12	fixed	I/O	Analog	Bidirectional differential data pin (D+ per USB specification).
USB0EPEN	K1 M1 L6 A11 D10	PG0 (7) PC5 (6) PA6 (8) PB2 (8) PH3 (4)	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
USB0ID	E12	PB0	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 the A side of the cable and pulled up is the B side).
USB0PFLT	L2 M2 M6 E11 B11 B10 B6	PC7 (6) PC6 (7) PA7 (8) PB3 (8) PE0 (9) PH4 (4) PJ1 (9)	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.

Table 19-2. Signals for USB (108BGA) (continued)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
USB0RBIAS	B12	fixed	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.
USB0VBUS	D12	PB1	I/O	Analog	This signal is used during the session request 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 has TTL-compatible voltage levels.

19.3 Functional Description

Note: A 9.1-kΩ resistor should be connected between the USB0RBIAS and ground. The 9.1-kΩ 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 USB controller provides full OTG negotiation by supporting both the session request protocol (SRP) and the host negotiation protocol (HNP). The session request protocol allows devices on the B side of a cable to request 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 and supports full OTG negotiations with other OTG devices.

In addition, the USB controller provides support for connecting to non-OTG peripherals or Host controllers. The USB controller can be configured to act as either a dedicated Host or Device, in which case, the USB0VBUS and USB0ID signals can be used as GPIOs. However, when the USB controller is acting as a self-powered Device, a GPIO input or analog comparator input must be connected to VBUS and configured to generate an interrupt when the VBUS level drops. This interrupt is used to disable the pullup resistor on the USB0DP signal.

Note: When USB is used in the system, the minimum system frequency is 20 MHz.

19.3.1 Operation as a Device

This section describes the Stellaris USB controller's actions when it is being used as a USB Device. Before the USB controller's operating mode is changed from Device to Host or Host to Device, software must reset the USB controller by setting the USB0 bit in the **Software Reset Control 2 (SRCR2)** register (see page 299). 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 the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used (described further in the following section).

- **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.

19.3.1.1 Endpoints

When operating as a Device, the USB controller provides two dedicated control endpoints (IN and OUT) and 30 configurable endpoints (15 IN and 15 OUT) that can be used for communications with a Host controller. The endpoint number and direction associated with an endpoint is directly related to its register designation. For example, when the Host is transmitting to endpoint 1, all configuration and data is in the endpoint 1 transmit 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 30 endpoints can be configured as control, bulk, interrupt, or isochronous endpoints. They should be treated as 15 configurable IN and 15 configurable OUT endpoints. The endpoint pairs 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.

19.3.1.2 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 15 configurable IN endpoints are determined by the **USB Transmit FIFO Start Address (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 **USB Maximum Transmit Data Endpoint n (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.

Note: The maximum packet size set for any endpoint must not exceed the FIFO size. The **USBTXMAXPn** register should not be written to while data is 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 **USB Transmit Dynamic FIFO Sizing (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 **USB Transmit Control and Status Endpoint n Low (USBTXCSR_Ln)** register must be set. If the AUTOSET bit in the **USB Transmit Control and Status Endpoint n High (USBTXCSR_Hn)** 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 must be set. If the AUTOSET bit in the **USBTXCSRn** 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 automatically 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 in the **USBTXCSRLn** register at this point indicates how many packets may be loaded. If the FIFONE bit is set, then another packet is in the FIFO and only one more packet can be loaded. If the FIFONE bit is clear, then no packets are 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 **USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS)** register. This bit is set by default, so it must be cleared to enable double-packet buffering.

19.3.1.3 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 15 configurable OUT endpoints are determined by the **USB Receive FIFO Start Address (USBRXFIFOADD)** register. The maximum amount of data received by an endpoint in any packet is determined by the value written to the **USB Maximum Receive Data Endpoint n (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.

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 **USB Receive Control and Status Endpoint n Low (USBRXCSRLn)** 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 must 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 **USB Receive Control and Status Endpoint n High (USBRXCSRn)** 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 **USBRXCSRLn** 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 **USBRXCSRLn** 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 must be cleared to allow further packets to be received. If the AUTOCL bit in the **USBRXCSRHn** 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 is set when RXRDY is cleared, the USB controller first clears the FULL bit, 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 **USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS)** register. This bit is set by default, so it must be cleared to enable double-packet buffering.

19.3.1.4 Scheduling

The Device has no control over the scheduling of transactions as scheduling is determined by the Host controller. The Stellaris USB controller can set up a transaction at any time. The USB controller waits for the request from the Host controller and generates 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.

19.3.1.5 Additional Actions

The USB controller responds automatically to certain conditions on the USB bus or actions by the Host controller such as when the USB controller automatically stalls a control transfer or 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 **USB Control and Status Endpoint 0 Low (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 DATAEND bit in the **USBCSRL0** register.

Setting the Device Address

When a Host 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 **USB Device Functional Address (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.

19.3.1.6 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 in the **USB Interrupt Enable (USBIE)** register, 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 **USB Power (USBPOWER)** register. When this bit is set, the USB controller exits SUSPEND mode and drives RESUME signaling onto the bus. The RESUME bit must be 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 mode which keeps the controller in a static state. The USB controller is not able to Hibernate because all the internal states are lost as a result.

19.3.1.7 Start-of-Frame

When the USB controller is operating in Device mode, it receives a Start-Of-Frame (SOF) 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 **USB Frame Value (USBFRAME)** 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, the packet is assumed to have been lost, and the **USBFRAME** register is not updated. The

USB controller continues and resynchronizes these pulses to the received SOF packets when these packets are successfully received again.

19.3.1.8 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 **USB Endpoint Index (USBEPIIDX)** 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, any open pipes are closed and the USB controller waits for bus enumeration to begin.

19.3.1.9 Connect/Disconnect

The USB controller connection to the USB bus is handled 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 the 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 does 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. The non-driving mode 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 bus. 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.

19.3.2 Operation as a Host

When the Stellaris 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. Before the USB controller's operating mode is changed from Host to Device or Device to Host, software must reset the USB controller by setting the USB0 bit in the **Software Reset Control 2 (SRCR2)** register (see page 299). 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 the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used (described further in the following section).
- **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.

19.3.2.1 Endpoints

The endpoint registers are used to control the USB endpoint interfaces which communicate with Device(s) that are connected. The endpoints consist of a dedicated control IN endpoint, a dedicated control OUT endpoint, 15 configurable OUT endpoints, and 15 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 15 independent OUT and 15 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 **USB Receive Functional Address Endpoint n (USBRXFUNCADDRn)** or **USB Transmit Functional Address Endpoint n (USBTXFUNCADDRn)** registers must 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. Customization includes allowing one FIFO per transaction, sharing a FIFO across transactions, and allowing for double-buffered FIFOs.

19.3.2.2 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 must be initiated by setting the **REQPKT** bit in the **USBCSRL0** register, indicating 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 the **USBCSRL0** register 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 must be cleared. The AUTOCL bit in the **USBRXCSR n** register can be used to have RXRDY automatically cleared when a maximum-sized packet has been unloaded from the FIFO. The AUTORQ bit in **USBRXCSR n** causes the REQPKT bit to be automatically set when the RXRDY bit is cleared. The AUTOCL and AUTORQ bits can be used with μ DMA accesses to perform complete bulk transfers without main processor intervention. When the RXRDY bit is cleared, the controller sends an acknowledge to the Device. When there is a known number of packets to be transferred, the **USB Request Packet Count in Block Transfer Endpoint n (USBRQPKTCOUNT n)** register associated with the endpoint should be configured to the number of packets to be transferred. The USB controller decrements the value in the **USBRQPKTCOUNT n** register following each request. When the **USBRQPKTCOUNT n** 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, **USBRQPKTCOUNT n** should be cleared. AUTORQ then remains set until cleared by the reception of a short packet (that is, less than the MAXLOAD value in the **USBRXMAXP n** register) 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 sets the STALLED bit in the **USBCSRL0** register. If the target Device does not respond to the IN token within the required time, or the packet contained a CRC or bit-stuff error, 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 sets the ERROR bit in the **USBCSRL0** register.

19.3.2.3 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 **USBTXCSRL n** register must be set as each packet is loaded into the transmit FIFO. Again, setting the AUTOSET bit in the **USBTXCSR n** register automatically sets TXRDY when a maximum-sized packet has been loaded into the FIFO. Furthermore, AUTOSET can be used with the μ 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 **USBTXCSRL n** register. If the target Device does not respond to the OUT token within the required time, or the packet contained a CRC or bit-stuff error, 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 sets the ERROR bit in the **USBTXCSRL n** register.

19.3.2.4 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 isochronous or interrupt transaction is started if the transaction is found on the first scheduler cycle of a frame and if the interval counter for that endpoint has counted down to zero. As a result, only one interrupt or isochronous transaction occurs per endpoint every n frames, where n is the interval set via the **USB Host Transmit Interval Endpoint n (USBTXINTERVALn)** or **USB Host Receive Interval Endpoint n (USBRXINTERVALn)** register for that endpoint.

An active bulk transaction starts immediately, provided sufficient time is left in the frame to complete the transaction before the next SOF packet is due. If the transaction must 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 process ensures that an endpoint that is sending a lot of NAKs does not block other transactions on the bus. The controller 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.

19.3.2.5 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 must be recorded in the corresponding **USB Receive Hub Address Endpoint n (USBRXHUBADDRn)** and **USB Receive Hub Port Endpoint n (USBRXHUBPORTn)** or the **USB Transmit Hub Address Endpoint n (USBTXHUBADDRn)** and **USB Transmit Hub Port Endpoint n (USBTXHUBPORTn)** registers. In addition, the speed at which the Device operates (full or low) must be recorded in the **USB Type Endpoint 0 (USBTYPE0)** (endpoint 0), **USB Host Configure Transmit Type Endpoint n (USBTXTYPEn)**, or **USB Host Configure Receive Type Endpoint n (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 must be made following the completion of any on-going transactions on the endpoints affected.

19.3.2.6 Babble

The USB Host controller does not start a transaction until the bus has been inactive for at least the minimum inter-packet delay. The controller 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.

19.3.2.7 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, set the **RESUME** bit and clear the **SUSPEND** bit. While the **RESUME** bit is set, the USB Host controller generates RESUME signaling on the bus. After 20 ms, the **RESUME** bit must be cleared, at which point the frame counter and transaction scheduler start. The Host supports the detection of a remote wake-up.

19.3.2.8 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 must 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.

19.3.2.9 Connect/Disconnect

A session is started by setting the **SESSION** bit in the **USB Device Control (USBDEVCTL)** register, enabling 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 set for a full-speed Device, and the **LSDEV** bit is set for a low-speed Device. The USB controller must 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.

19.3.3 OTG Mode

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). Note that when switching between OTG A and OTG B, the USB controller retains all register contents.

19.3.3.1 Starting a Session

When the USB OTG controller is ready to start a session, the **SESSION** bit must 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. The USB OTG controller also provides an interrupt to indicate that ID pin sensing has completed and the mode value in the **USBDEVCTL** register is valid. This interrupt is enabled in the **USBIDVIM** register, and the status is checked in the **USBIDVISC** register. As soon as the USB controller has detected that it is on the A side of the cable, it must enable VBUS power within 100ms or the USB controller reverts to Device mode.

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 can be cleared to end a session. The USB OTG controller also automatically ends the session if babble is detected or if VBUS drops below session valid.

Note: The USB OTG controller may not remain in Host mode when connected to high-current devices. Some devices draw enough current to momentarily drop VBUS below the VBUS-valid level causing the controller to drop out of Host mode. The only way to get back into Host mode is to allow VBUS to go below the Session End level. In this situation, the device is causing VBUS to drop repeatedly and pull VBUS back low the next time VBUS is enabled.

In addition, the USB OTG controller may not remain in Host mode when a device is told that it can start using its active configuration. At this point the device starts drawing more current and can also drop VBUS below VBUS valid.

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 first discharges 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+, signaling the A device to end the session.

19.3.3.2 Detecting Activity

When the other device of the OTG setup 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 setup 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 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 drop and clears the SESSION bit to indicate that the session has ended, causing a disconnect interrupt to be signaled. If data line and VBUS pulsing is detected, then the USB controller is the A device. The controller generates a SESSION REQUEST interrupt to indicate that the B device is requesting a session. The SESSION bit in the **USBDEVCTL** register must be set to start a session.

19.3.3.3 Host Negotiation

When the USB controller is the A device, ID is Low, and the controller automatically enters Host mode when a session starts. When the USB controller is the B device, ID is High, and the controller automatically enters Device mode when a session starts. However, software 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 and if the HOSTREQ bit remains set, the controller 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, causing the A device to switch to Device mode and connect its own pull-up resistor. When the USB controller detects this, a Connect interrupt is generated and the RESET bit in the **USBPOWER** register is set 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, the USB controller 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.

19.3.4 DMA Operation

The USB peripheral provides an interface connected to the µDMA controller with separate channels for 3 transmit endpoints and 3 receive endpoints. Software selects which endpoints to service with the µDMA channels using the **USB DMA Select (USBDMASEL)** register. The µDMA operation of the USB is enabled through the **USBTXCSR n** and **USBRXCSR n** registers, for the TX and RX channels respectively. When µDMA operation is enabled, the USB asserts a µDMA 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 µDMA channel must be configured to operate in Basic mode, and the size of the µDMA transfer must be restricted to whole multiples of the size of the USB FIFO. Both read and write transfers of the USB FIFOs using µDMA must be configured in this manner. For example, if the USB endpoint is configured with a FIFO size of 64 bytes, the µDMA 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 the **DMAMOD** bit in the **USBTXCSR n /USBRXCSR n** register is clear, an interrupt is generated after every packet is transferred, but the µDMA continues transferring data. If the **DMAMOD** bit is set, an interrupt is generated only when the entire µDMA transfer is complete. The interrupt occurs on the USB interrupt vector. Therefore, if interrupts are used for USB operation and the µDMA is enabled, the USB interrupt handler must be designed to handle the µDMA completion interrupt.

Care must be taken when using the µDMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of value of the **MAXLOAD** field in the **USBRXCSR n** register. The **RXRDY** bit is cleared as follows.

Table 19-3. Remainder (MAXLOAD/4)

Value	Description
0	MAXLOAD = 64 bytes
1	MAXLOAD = 61 bytes
2	MAXLOAD = 62 bytes
3	MAXLOAD = 63 bytes

Table 19-4. Actual Bytes Read

Value	Description
0	MAXLOAD
1	MAXLOAD+3
2	MAXLOAD+2
3	MAXLOAD+1

Table 19-5. Packet Sizes That Clear RXRDY

Value	Description
0	MAXLOAD, MAXLOAD-1, MAXLOAD-2, MAXLOAD-3
1	MAXLOAD
2	MAXLOAD, MAXLOAD-1
3	MAXLOAD, MAXLOAD-1, MAXLOAD-2

To enable DMA operation for the endpoint receive channel, the `DMAEN` bit of the **USBRXCSR_n** register should be set. To enable DMA operation for the endpoint transmit channel, the `DMAEN` bit of the **USBTXCSR_n** register must be set.

See “Micro Direct Memory Access (μDMA)” on page 365 for more details about programming the μDMA controller.

19.4 Initialization and Configuration

To use the USB Controller, the peripheral clock must be enabled via the **RCGC2** register (see page 285). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module (see page 285). To find out which GPIO port to enable, refer to Table 24-4 on page 1182. Configure the `PMCn` fields in the **GPIOPCTL** register to assign the USB signals to the appropriate pins (see page 465 and Table 24-5 on page 1190).

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 negated on start up by configuring the `USB0EPEN` and `USB0PFLT` pins to be controlled by the USB controller and not exhibit their default GPIO behavior.

Note: When used in OTG mode, `USB0VBUS` and `USB0ID` do not require any configuration as they are dedicated pins for the USB controller and directly connect to the USB connector’s VBUS and ID signals. If the USB controller is used as either a dedicated Host or Device, the `DEVMODOTG` and `DEVMOD` bits in the **USB General-Purpose Control and Status (USBGPCS)** register can be used to connect the `USB0VBUS` and `USB0ID` inputs to fixed levels internally, freeing the `PB0` and `PB1` pins for GPIO use. For proper self-powered Device operation, the VBUS value must still be monitored to assure that if the Host removes VBUS, the self-powered Device disables the D+/D- pull-up resistors. This function can be accomplished by connecting a standard GPIO to VBUS.

19.4.1 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 negated 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. An input pin, `USB0PFLT`, provides feedback when there has been a power fault on VBUS. The `USB0PFLT` signal can be configured to either automatically negate the `USB0EPEN` signal to disable power, and/or it can generate an interrupt to the interrupt controller to allow software 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.

19.4.2 Endpoint Configuration

To start communication in Host or Device mode, the endpoint registers must first be configured. In Host mode, this configuration establishes a connection between an endpoint register and an endpoint on a Device. In Device mode, an endpoint must be configured before enumerating to the Host controller.

In both cases, the endpoint 0 configuration is limited because it 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 reserved for endpoint 0. The endpoint's FIFO must be at least as large as 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 controller's soft connect must be enabled when the Device is ready to start communications, indicating to the Host controller that the Device is ready to start the enumeration process. If operating as a Host controller, the Device soft connect must be disabled and power must be provided to VBUS via the `USBOEPEN` signal.

19.5 Register Map

Table 19-6 on page 930 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 285).

Table 19-6. Universal Serial Bus (USB) Controller Register Map

Offset	Name	Type	Reset	Description	See page
0x000	USBFADDR	R/W	0x00	USB Device Functional Address	942
0x001	USBPOWER	R/W	0x20	USB Power	943
0x002	USBTXIS	RO	0x0000	USB Transmit Interrupt Status	946
0x004	USBRXIS	RO	0x0000	USB Receive Interrupt Status	948
0x006	USBTXIE	R/W	0xFFFF	USB Transmit Interrupt Enable	950
0x008	USBRXIE	R/W	0xFFFFE	USB Receive Interrupt Enable	952
0x00A	USBIS	RO	0x00	USB General Interrupt Status	954
0x00B	USBIE	R/W	0x06	USB Interrupt Enable	957
0x00C	USBFRAME	RO	0x0000	USB Frame Value	960
0x00E	USBEPIDX	R/W	0x00	USB Endpoint Index	961
0x00F	USBTEST	R/W	0x00	USB Test Mode	962
0x020	USBFIFO0	R/W	0x0000.0000	USB FIFO Endpoint 0	964
0x024	USBFIFO1	R/W	0x0000.0000	USB FIFO Endpoint 1	964
0x028	USBFIFO2	R/W	0x0000.0000	USB FIFO Endpoint 2	964

Table 19-6. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x02C	USBFIFO3	R/W	0x0000.0000	USB FIFO Endpoint 3	964
0x030	USBFIFO4	R/W	0x0000.0000	USB FIFO Endpoint 4	964
0x034	USBFIFO5	R/W	0x0000.0000	USB FIFO Endpoint 5	964
0x038	USBFIFO6	R/W	0x0000.0000	USB FIFO Endpoint 6	964
0x03C	USBFIFO7	R/W	0x0000.0000	USB FIFO Endpoint 7	964
0x040	USBFIFO8	R/W	0x0000.0000	USB FIFO Endpoint 8	964
0x044	USBFIFO9	R/W	0x0000.0000	USB FIFO Endpoint 9	964
0x048	USBFIFO10	R/W	0x0000.0000	USB FIFO Endpoint 10	964
0x04C	USBFIFO11	R/W	0x0000.0000	USB FIFO Endpoint 11	964
0x050	USBFIFO12	R/W	0x0000.0000	USB FIFO Endpoint 12	964
0x054	USBFIFO13	R/W	0x0000.0000	USB FIFO Endpoint 13	964
0x058	USBFIFO14	R/W	0x0000.0000	USB FIFO Endpoint 14	964
0x05C	USBFIFO15	R/W	0x0000.0000	USB FIFO Endpoint 15	964
0x060	USBDEVCTL	R/W	0x80	USB Device Control	966
0x062	USBTXFIFOSZ	R/W	0x00	USB Transmit Dynamic FIFO Sizing	968
0x063	USBRXFIFOSZ	R/W	0x00	USB Receive Dynamic FIFO Sizing	968
0x064	USBTXFIFOADD	R/W	0x0000	USB Transmit FIFO Start Address	969
0x066	USBRXFIFOADD	R/W	0x0000	USB Receive FIFO Start Address	969
0x07A	USBCONTIM	R/W	0x5C	USB Connect Timing	970
0x07B	USBVPLEN	R/W	0x3C	USB OTG VBUS Pulse Timing	971
0x07D	USBFSEOF	R/W	0x77	USB Full-Speed Last Transaction to End of Frame Timing	972
0x07E	USBLSEOF	R/W	0x72	USB Low-Speed Last Transaction to End of Frame Timing	973
0x080	USBTXFUNCADDR0	R/W	0x00	USB Transmit Functional Address Endpoint 0	974
0x082	USBTXHUBADDR0	R/W	0x00	USB Transmit Hub Address Endpoint 0	976
0x083	USBTXHUBPORT0	R/W	0x00	USB Transmit Hub Port Endpoint 0	978
0x088	USBTXFUNCADDR1	R/W	0x00	USB Transmit Functional Address Endpoint 1	974
0x08A	USBTXHUBADDR1	R/W	0x00	USB Transmit Hub Address Endpoint 1	976
0x08B	USBTXHUBPORT1	R/W	0x00	USB Transmit Hub Port Endpoint 1	978
0x08C	USBRXFUNCADDR1	R/W	0x00	USB Receive Functional Address Endpoint 1	980
0x08E	USBRXHUBADDR1	R/W	0x00	USB Receive Hub Address Endpoint 1	982
0x08F	USBRXHUBPORT1	R/W	0x00	USB Receive Hub Port Endpoint 1	984

Table 19-6. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x090	USBTXFUNCADDR2	R/W	0x00	USB Transmit Functional Address Endpoint 2	974
0x092	USBTXHUBADDR2	R/W	0x00	USB Transmit Hub Address Endpoint 2	976
0x093	USBTXHUBPORT2	R/W	0x00	USB Transmit Hub Port Endpoint 2	978
0x094	USBRXFUNCADDR2	R/W	0x00	USB Receive Functional Address Endpoint 2	980
0x096	USBRXHUBADDR2	R/W	0x00	USB Receive Hub Address Endpoint 2	982
0x097	USBRXHUBPORT2	R/W	0x00	USB Receive Hub Port Endpoint 2	984
0x098	USBTXFUNCADDR3	R/W	0x00	USB Transmit Functional Address Endpoint 3	974
0x09A	USBTXHUBADDR3	R/W	0x00	USB Transmit Hub Address Endpoint 3	976
0x09B	USBTXHUBPORT3	R/W	0x00	USB Transmit Hub Port Endpoint 3	978
0x09C	USBRXFUNCADDR3	R/W	0x00	USB Receive Functional Address Endpoint 3	980
0x09E	USBRXHUBADDR3	R/W	0x00	USB Receive Hub Address Endpoint 3	982
0x09F	USBRXHUBPORT3	R/W	0x00	USB Receive Hub Port Endpoint 3	984
0x0A0	USBTXFUNCADDR4	R/W	0x00	USB Transmit Functional Address Endpoint 4	974
0x0A2	USBTXHUBADDR4	R/W	0x00	USB Transmit Hub Address Endpoint 4	976
0x0A3	USBTXHUBPORT4	R/W	0x00	USB Transmit Hub Port Endpoint 4	978
0x0A4	USBRXFUNCADDR4	R/W	0x00	USB Receive Functional Address Endpoint 4	980
0x0A6	USBRXHUBADDR4	R/W	0x00	USB Receive Hub Address Endpoint 4	982
0x0A7	USBRXHUBPORT4	R/W	0x00	USB Receive Hub Port Endpoint 4	984
0x0A8	USBTXFUNCADDR5	R/W	0x00	USB Transmit Functional Address Endpoint 5	974
0x0AA	USBTXHUBADDR5	R/W	0x00	USB Transmit Hub Address Endpoint 5	976
0x0AB	USBTXHUBPORT5	R/W	0x00	USB Transmit Hub Port Endpoint 5	978
0x0AC	USBRXFUNCADDR5	R/W	0x00	USB Receive Functional Address Endpoint 5	980
0x0AE	USBRXHUBADDR5	R/W	0x00	USB Receive Hub Address Endpoint 5	982
0x0AF	USBRXHUBPORT5	R/W	0x00	USB Receive Hub Port Endpoint 5	984
0x0B0	USBTXFUNCADDR6	R/W	0x00	USB Transmit Functional Address Endpoint 6	974
0x0B2	USBTXHUBADDR6	R/W	0x00	USB Transmit Hub Address Endpoint 6	976
0x0B3	USBTXHUBPORT6	R/W	0x00	USB Transmit Hub Port Endpoint 6	978
0x0B4	USBRXFUNCADDR6	R/W	0x00	USB Receive Functional Address Endpoint 6	980
0x0B6	USBRXHUBADDR6	R/W	0x00	USB Receive Hub Address Endpoint 6	982
0x0B7	USBRXHUBPORT6	R/W	0x00	USB Receive Hub Port Endpoint 6	984
0x0B8	USBTXFUNCADDR7	R/W	0x00	USB Transmit Functional Address Endpoint 7	974
0x0BA	USBTXHUBADDR7	R/W	0x00	USB Transmit Hub Address Endpoint 7	976

Table 19-6. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x0BB	USBTXHUBPORT7	R/W	0x00	USB Transmit Hub Port Endpoint 7	978
0x0BC	USBRXFUNCADDR7	R/W	0x00	USB Receive Functional Address Endpoint 7	980
0x0BE	USBRXHUBADDR7	R/W	0x00	USB Receive Hub Address Endpoint 7	982
0x0BF	USBRXHUBPORT7	R/W	0x00	USB Receive Hub Port Endpoint 7	984
0x0C0	USBTXFUNCADDR8	R/W	0x00	USB Transmit Functional Address Endpoint 8	974
0x0C2	USBTXHUBADDR8	R/W	0x00	USB Transmit Hub Address Endpoint 8	976
0x0C3	USBTXHUBPORT8	R/W	0x00	USB Transmit Hub Port Endpoint 8	978
0x0C4	USBRXFUNCADDR8	R/W	0x00	USB Receive Functional Address Endpoint 8	980
0x0C6	USBRXHUBADDR8	R/W	0x00	USB Receive Hub Address Endpoint 8	982
0x0C7	USBRXHUBPORT8	R/W	0x00	USB Receive Hub Port Endpoint 8	984
0x0C8	USBTXFUNCADDR9	R/W	0x00	USB Transmit Functional Address Endpoint 9	974
0x0CA	USBTXHUBADDR9	R/W	0x00	USB Transmit Hub Address Endpoint 9	976
0x0CB	USBTXHUBPORT9	R/W	0x00	USB Transmit Hub Port Endpoint 9	978
0x0CC	USBRXFUNCADDR9	R/W	0x00	USB Receive Functional Address Endpoint 9	980
0x0CE	USBRXHUBADDR9	R/W	0x00	USB Receive Hub Address Endpoint 9	982
0x0CF	USBRXHUBPORT9	R/W	0x00	USB Receive Hub Port Endpoint 9	984
0x0D0	USBTXFUNCADDR10	R/W	0x00	USB Transmit Functional Address Endpoint 10	974
0x0D2	USBTXHUBADDR10	R/W	0x00	USB Transmit Hub Address Endpoint 10	976
0x0D3	USBTXHUBPORT10	R/W	0x00	USB Transmit Hub Port Endpoint 10	978
0x0D4	USBRXFUNCADDR10	R/W	0x00	USB Receive Functional Address Endpoint 10	980
0x0D6	USBRXHUBADDR10	R/W	0x00	USB Receive Hub Address Endpoint 10	982
0x0D7	USBRXHUBPORT10	R/W	0x00	USB Receive Hub Port Endpoint 10	984
0x0D8	USBTXFUNCADDR11	R/W	0x00	USB Transmit Functional Address Endpoint 11	974
0x0DA	USBTXHUBADDR11	R/W	0x00	USB Transmit Hub Address Endpoint 11	976
0x0DB	USBTXHUBPORT11	R/W	0x00	USB Transmit Hub Port Endpoint 11	978
0x0DC	USBRXFUNCADDR11	R/W	0x00	USB Receive Functional Address Endpoint 11	980
0x0DE	USBRXHUBADDR11	R/W	0x00	USB Receive Hub Address Endpoint 11	982
0x0DF	USBRXHUBPORT11	R/W	0x00	USB Receive Hub Port Endpoint 11	984
0x0E0	USBTXFUNCADDR12	R/W	0x00	USB Transmit Functional Address Endpoint 12	974
0x0E2	USBTXHUBADDR12	R/W	0x00	USB Transmit Hub Address Endpoint 12	976
0x0E3	USBTXHUBPORT12	R/W	0x00	USB Transmit Hub Port Endpoint 12	978
0x0E4	USBRXFUNCADDR12	R/W	0x00	USB Receive Functional Address Endpoint 12	980

Table 19-6. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x0E6	USBRXHUBADDR12	R/W	0x00	USB Receive Hub Address Endpoint 12	982
0x0E7	USBRXHUBPORT12	R/W	0x00	USB Receive Hub Port Endpoint 12	984
0x0E8	USBTXFUNCADDR13	R/W	0x00	USB Transmit Functional Address Endpoint 13	974
0x0EA	USBTXHUBADDR13	R/W	0x00	USB Transmit Hub Address Endpoint 13	976
0x0EB	USBTXHUBPORT13	R/W	0x00	USB Transmit Hub Port Endpoint 13	978
0x0EC	USBRXFUNCADDR13	R/W	0x00	USB Receive Functional Address Endpoint 13	980
0x0EE	USBRXHUBADDR13	R/W	0x00	USB Receive Hub Address Endpoint 13	982
0x0EF	USBRXHUBPORT13	R/W	0x00	USB Receive Hub Port Endpoint 13	984
0x0F0	USBTXFUNCADDR14	R/W	0x00	USB Transmit Functional Address Endpoint 14	974
0x0F2	USBTXHUBADDR14	R/W	0x00	USB Transmit Hub Address Endpoint 14	976
0x0F3	USBTXHUBPORT14	R/W	0x00	USB Transmit Hub Port Endpoint 14	978
0x0F4	USBRXFUNCADDR14	R/W	0x00	USB Receive Functional Address Endpoint 14	980
0x0F6	USBRXHUBADDR14	R/W	0x00	USB Receive Hub Address Endpoint 14	982
0x0F7	USBRXHUBPORT14	R/W	0x00	USB Receive Hub Port Endpoint 14	984
0x0F8	USBTXFUNCADDR15	R/W	0x00	USB Transmit Functional Address Endpoint 15	974
0x0FA	USBTXHUBADDR15	R/W	0x00	USB Transmit Hub Address Endpoint 15	976
0x0FB	USBTXHUBPORT15	R/W	0x00	USB Transmit Hub Port Endpoint 15	978
0x0FC	USBRXFUNCADDR15	R/W	0x00	USB Receive Functional Address Endpoint 15	980
0x0FE	USBRXHUBADDR15	R/W	0x00	USB Receive Hub Address Endpoint 15	982
0x0FF	USBRXHUBPORT15	R/W	0x00	USB Receive Hub Port Endpoint 15	984
0x102	USBCSRL0	W1C	0x00	USB Control and Status Endpoint 0 Low	988
0x103	USBCSRH0	W1C	0x00	USB Control and Status Endpoint 0 High	992
0x108	USBCOUNT0	RO	0x00	USB Receive Byte Count Endpoint 0	994
0x10A	USBTYPE0	R/W	0x00	USB Type Endpoint 0	995
0x10B	USBNAKLMT	R/W	0x00	USB NAK Limit	996
0x110	USBTXMAXP1	R/W	0x0000	USB Maximum Transmit Data Endpoint 1	986
0x112	USBTXCSR1	R/W	0x00	USB Transmit Control and Status Endpoint 1 Low	997
0x113	USBTXCSR1H	R/W	0x00	USB Transmit Control and Status Endpoint 1 High	1002
0x114	USBRXMAXP1	R/W	0x0000	USB Maximum Receive Data Endpoint 1	1006
0x116	USBRXCSR1	R/W	0x00	USB Receive Control and Status Endpoint 1 Low	1008
0x117	USBRXCSR1H	R/W	0x00	USB Receive Control and Status Endpoint 1 High	1013
0x118	USBRXCOUNT1	RO	0x0000	USB Receive Byte Count Endpoint 1	1018

Table 19-6. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x11A	USBTXTYPE1	R/W	0x00	USB Host Transmit Configure Type Endpoint 1	1020
0x11B	USBTXINTERVAL1	R/W	0x00	USB Host Transmit Interval Endpoint 1	1022
0x11C	USBRXTYPE1	R/W	0x00	USB Host Configure Receive Type Endpoint 1	1024
0x11D	USBRXINTERVAL1	R/W	0x00	USB Host Receive Polling Interval Endpoint 1	1026
0x120	USBTXMAXP2	R/W	0x0000	USB Maximum Transmit Data Endpoint 2	986
0x122	USBTXCSRL2	R/W	0x00	USB Transmit Control and Status Endpoint 2 Low	997
0x123	USBTXCSRH2	R/W	0x00	USB Transmit Control and Status Endpoint 2 High	1002
0x124	USBRXMAXP2	R/W	0x0000	USB Maximum Receive Data Endpoint 2	1006
0x126	USBRXCSRL2	R/W	0x00	USB Receive Control and Status Endpoint 2 Low	1008
0x127	USBRXCSRH2	R/W	0x00	USB Receive Control and Status Endpoint 2 High	1013
0x128	USBRXCOUNT2	RO	0x0000	USB Receive Byte Count Endpoint 2	1018
0x12A	USBTXTYPE2	R/W	0x00	USB Host Transmit Configure Type Endpoint 2	1020
0x12B	USBTXINTERVAL2	R/W	0x00	USB Host Transmit Interval Endpoint 2	1022
0x12C	USBRXTYPE2	R/W	0x00	USB Host Configure Receive Type Endpoint 2	1024
0x12D	USBRXINTERVAL2	R/W	0x00	USB Host Receive Polling Interval Endpoint 2	1026
0x130	USBTXMAXP3	R/W	0x0000	USB Maximum Transmit Data Endpoint 3	986
0x132	USBTXCSRL3	R/W	0x00	USB Transmit Control and Status Endpoint 3 Low	997
0x133	USBTXCSRH3	R/W	0x00	USB Transmit Control and Status Endpoint 3 High	1002
0x134	USBRXMAXP3	R/W	0x0000	USB Maximum Receive Data Endpoint 3	1006
0x136	USBRXCSRL3	R/W	0x00	USB Receive Control and Status Endpoint 3 Low	1008
0x137	USBRXCSRH3	R/W	0x00	USB Receive Control and Status Endpoint 3 High	1013
0x138	USBRXCOUNT3	RO	0x0000	USB Receive Byte Count Endpoint 3	1018
0x13A	USBTXTYPE3	R/W	0x00	USB Host Transmit Configure Type Endpoint 3	1020
0x13B	USBTXINTERVAL3	R/W	0x00	USB Host Transmit Interval Endpoint 3	1022
0x13C	USBRXTYPE3	R/W	0x00	USB Host Configure Receive Type Endpoint 3	1024
0x13D	USBRXINTERVAL3	R/W	0x00	USB Host Receive Polling Interval Endpoint 3	1026
0x140	USBTXMAXP4	R/W	0x0000	USB Maximum Transmit Data Endpoint 4	986
0x142	USBTXCSRL4	R/W	0x00	USB Transmit Control and Status Endpoint 4 Low	997
0x143	USBTXCSRH4	R/W	0x00	USB Transmit Control and Status Endpoint 4 High	1002
0x144	USBRXMAXP4	R/W	0x0000	USB Maximum Receive Data Endpoint 4	1006
0x146	USBRXCSRL4	R/W	0x00	USB Receive Control and Status Endpoint 4 Low	1008
0x147	USBRXCSRH4	R/W	0x00	USB Receive Control and Status Endpoint 4 High	1013

Table 19-6. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x148	USBRXCOUNT4	RO	0x0000	USB Receive Byte Count Endpoint 4	1018
0x14A	USBTXTYPE4	R/W	0x00	USB Host Transmit Configure Type Endpoint 4	1020
0x14B	USBTXINTERVAL4	R/W	0x00	USB Host Transmit Interval Endpoint 4	1022
0x14C	USBRXTYPE4	R/W	0x00	USB Host Configure Receive Type Endpoint 4	1024
0x14D	USBRXINTERVAL4	R/W	0x00	USB Host Receive Polling Interval Endpoint 4	1026
0x150	USBTXMAXP5	R/W	0x0000	USB Maximum Transmit Data Endpoint 5	986
0x152	USBTXCSRL5	R/W	0x00	USB Transmit Control and Status Endpoint 5 Low	997
0x153	USBTXCSRH5	R/W	0x00	USB Transmit Control and Status Endpoint 5 High	1002
0x154	USBRXMAXP5	R/W	0x0000	USB Maximum Receive Data Endpoint 5	1006
0x156	USBRXCSRL5	R/W	0x00	USB Receive Control and Status Endpoint 5 Low	1008
0x157	USBRXCSRH5	R/W	0x00	USB Receive Control and Status Endpoint 5 High	1013
0x158	USBRXCOUNT5	RO	0x0000	USB Receive Byte Count Endpoint 5	1018
0x15A	USBTXTYPE5	R/W	0x00	USB Host Transmit Configure Type Endpoint 5	1020
0x15B	USBTXINTERVAL5	R/W	0x00	USB Host Transmit Interval Endpoint 5	1022
0x15C	USBRXTYPE5	R/W	0x00	USB Host Configure Receive Type Endpoint 5	1024
0x15D	USBRXINTERVAL5	R/W	0x00	USB Host Receive Polling Interval Endpoint 5	1026
0x160	USBTXMAXP6	R/W	0x0000	USB Maximum Transmit Data Endpoint 6	986
0x162	USBTXCSRL6	R/W	0x00	USB Transmit Control and Status Endpoint 6 Low	997
0x163	USBTXCSRH6	R/W	0x00	USB Transmit Control and Status Endpoint 6 High	1002
0x164	USBRXMAXP6	R/W	0x0000	USB Maximum Receive Data Endpoint 6	1006
0x166	USBRXCSRL6	R/W	0x00	USB Receive Control and Status Endpoint 6 Low	1008
0x167	USBRXCSRH6	R/W	0x00	USB Receive Control and Status Endpoint 6 High	1013
0x168	USBRXCOUNT6	RO	0x0000	USB Receive Byte Count Endpoint 6	1018
0x16A	USBTXTYPE6	R/W	0x00	USB Host Transmit Configure Type Endpoint 6	1020
0x16B	USBTXINTERVAL6	R/W	0x00	USB Host Transmit Interval Endpoint 6	1022
0x16C	USBRXTYPE6	R/W	0x00	USB Host Configure Receive Type Endpoint 6	1024
0x16D	USBRXINTERVAL6	R/W	0x00	USB Host Receive Polling Interval Endpoint 6	1026
0x170	USBTXMAXP7	R/W	0x0000	USB Maximum Transmit Data Endpoint 7	986
0x172	USBTXCSRL7	R/W	0x00	USB Transmit Control and Status Endpoint 7 Low	997
0x173	USBTXCSRH7	R/W	0x00	USB Transmit Control and Status Endpoint 7 High	1002
0x174	USBRXMAXP7	R/W	0x0000	USB Maximum Receive Data Endpoint 7	1006
0x176	USBRXCSRL7	R/W	0x00	USB Receive Control and Status Endpoint 7 Low	1008

Table 19-6. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x177	USBRXCSR7	R/W	0x00	USB Receive Control and Status Endpoint 7 High	1013
0x178	USBRXCOUNT7	RO	0x0000	USB Receive Byte Count Endpoint 7	1018
0x17A	USBTXTYPE7	R/W	0x00	USB Host Transmit Configure Type Endpoint 7	1020
0x17B	USBTXINTERVAL7	R/W	0x00	USB Host Transmit Interval Endpoint 7	1022
0x17C	USBRXTYPE7	R/W	0x00	USB Host Configure Receive Type Endpoint 7	1024
0x17D	USBRXINTERVAL7	R/W	0x00	USB Host Receive Polling Interval Endpoint 7	1026
0x180	USBTXMAXP8	R/W	0x0000	USB Maximum Transmit Data Endpoint 8	986
0x182	USBTXCSRL8	R/W	0x00	USB Transmit Control and Status Endpoint 8 Low	997
0x183	USBTXCSR8	R/W	0x00	USB Transmit Control and Status Endpoint 8 High	1002
0x184	USBRXMAXP8	R/W	0x0000	USB Maximum Receive Data Endpoint 8	1006
0x186	USBRXCSRL8	R/W	0x00	USB Receive Control and Status Endpoint 8 Low	1008
0x187	USBRXCSR8	R/W	0x00	USB Receive Control and Status Endpoint 8 High	1013
0x188	USBRXCOUNT8	RO	0x0000	USB Receive Byte Count Endpoint 8	1018
0x18A	USBTXTYPE8	R/W	0x00	USB Host Transmit Configure Type Endpoint 8	1020
0x18B	USBTXINTERVAL8	R/W	0x00	USB Host Transmit Interval Endpoint 8	1022
0x18C	USBRXTYPE8	R/W	0x00	USB Host Configure Receive Type Endpoint 8	1024
0x18D	USBRXINTERVAL8	R/W	0x00	USB Host Receive Polling Interval Endpoint 8	1026
0x190	USBTXMAXP9	R/W	0x0000	USB Maximum Transmit Data Endpoint 9	986
0x192	USBTXCSRL9	R/W	0x00	USB Transmit Control and Status Endpoint 9 Low	997
0x193	USBTXCSR9	R/W	0x00	USB Transmit Control and Status Endpoint 9 High	1002
0x194	USBRXMAXP9	R/W	0x0000	USB Maximum Receive Data Endpoint 9	1006
0x196	USBRXCSRL9	R/W	0x00	USB Receive Control and Status Endpoint 9 Low	1008
0x197	USBRXCSR9	R/W	0x00	USB Receive Control and Status Endpoint 9 High	1013
0x198	USBRXCOUNT9	RO	0x0000	USB Receive Byte Count Endpoint 9	1018
0x19A	USBTXTYPE9	R/W	0x00	USB Host Transmit Configure Type Endpoint 9	1020
0x19B	USBTXINTERVAL9	R/W	0x00	USB Host Transmit Interval Endpoint 9	1022
0x19C	USBRXTYPE9	R/W	0x00	USB Host Configure Receive Type Endpoint 9	1024
0x19D	USBRXINTERVAL9	R/W	0x00	USB Host Receive Polling Interval Endpoint 9	1026
0x1A0	USBTXMAXP10	R/W	0x0000	USB Maximum Transmit Data Endpoint 10	986
0x1A2	USBTXCSRL10	R/W	0x00	USB Transmit Control and Status Endpoint 10 Low	997
0x1A3	USBTXCSR10	R/W	0x00	USB Transmit Control and Status Endpoint 10 High	1002
0x1A4	USBRXMAXP10	R/W	0x0000	USB Maximum Receive Data Endpoint 10	1006

Table 19-6. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x1A6	USBRXCSR10	R/W	0x00	USB Receive Control and Status Endpoint 10 Low	1008
0x1A7	USBRXCSRH10	R/W	0x00	USB Receive Control and Status Endpoint 10 High	1013
0x1A8	USBRXCOUNT10	RO	0x0000	USB Receive Byte Count Endpoint 10	1018
0x1AA	USBTYPE10	R/W	0x00	USB Host Transmit Configure Type Endpoint 10	1020
0x1AB	USBTXINTERVAL10	R/W	0x00	USB Host Transmit Interval Endpoint 10	1022
0x1AC	USBRXTYPE10	R/W	0x00	USB Host Configure Receive Type Endpoint 10	1024
0x1AD	USBRXINTERVAL10	R/W	0x00	USB Host Receive Polling Interval Endpoint 10	1026
0x1B0	USBTXMAXP11	R/W	0x0000	USB Maximum Transmit Data Endpoint 11	986
0x1B2	USBTXCSR11	R/W	0x00	USB Transmit Control and Status Endpoint 11 Low	997
0x1B3	USBTXCSRH11	R/W	0x00	USB Transmit Control and Status Endpoint 11 High	1002
0x1B4	USBRXMAXP11	R/W	0x0000	USB Maximum Receive Data Endpoint 11	1006
0x1B6	USBRXCSR11	R/W	0x00	USB Receive Control and Status Endpoint 11 Low	1008
0x1B7	USBRXCSRH11	R/W	0x00	USB Receive Control and Status Endpoint 11 High	1013
0x1B8	USBRXCOUNT11	RO	0x0000	USB Receive Byte Count Endpoint 11	1018
0x1BA	USBTYPE11	R/W	0x00	USB Host Transmit Configure Type Endpoint 11	1020
0x1BB	USBTXINTERVAL11	R/W	0x00	USB Host Transmit Interval Endpoint 11	1022
0x1BC	USBRXTYPE11	R/W	0x00	USB Host Configure Receive Type Endpoint 11	1024
0x1BD	USBRXINTERVAL11	R/W	0x00	USB Host Receive Polling Interval Endpoint 11	1026
0x1C0	USBTXMAXP12	R/W	0x0000	USB Maximum Transmit Data Endpoint 12	986
0x1C2	USBTXCSR12	R/W	0x00	USB Transmit Control and Status Endpoint 12 Low	997
0x1C3	USBTXCSRH12	R/W	0x00	USB Transmit Control and Status Endpoint 12 High	1002
0x1C4	USBRXMAXP12	R/W	0x0000	USB Maximum Receive Data Endpoint 12	1006
0x1C6	USBRXCSR12	R/W	0x00	USB Receive Control and Status Endpoint 12 Low	1008
0x1C7	USBRXCSRH12	R/W	0x00	USB Receive Control and Status Endpoint 12 High	1013
0x1C8	USBRXCOUNT12	RO	0x0000	USB Receive Byte Count Endpoint 12	1018
0x1CA	USBTYPE12	R/W	0x00	USB Host Transmit Configure Type Endpoint 12	1020
0x1CB	USBTXINTERVAL12	R/W	0x00	USB Host Transmit Interval Endpoint 12	1022
0x1CC	USBRXTYPE12	R/W	0x00	USB Host Configure Receive Type Endpoint 12	1024
0x1CD	USBRXINTERVAL12	R/W	0x00	USB Host Receive Polling Interval Endpoint 12	1026
0x1D0	USBTXMAXP13	R/W	0x0000	USB Maximum Transmit Data Endpoint 13	986
0x1D2	USBTXCSR13	R/W	0x00	USB Transmit Control and Status Endpoint 13 Low	997
0x1D3	USBTXCSRH13	R/W	0x00	USB Transmit Control and Status Endpoint 13 High	1002

Table 19-6. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x1D4	USBRXMAXP13	R/W	0x0000	USB Maximum Receive Data Endpoint 13	1006
0x1D6	USBRXCSRL13	R/W	0x00	USB Receive Control and Status Endpoint 13 Low	1008
0x1D7	USBRXCSRH13	R/W	0x00	USB Receive Control and Status Endpoint 13 High	1013
0x1D8	USBRXCOUNT13	RO	0x0000	USB Receive Byte Count Endpoint 13	1018
0x1DA	USBTXTYPE13	R/W	0x00	USB Host Transmit Configure Type Endpoint 13	1020
0x1DB	USBTXINTERVAL13	R/W	0x00	USB Host Transmit Interval Endpoint 13	1022
0x1DC	USBRXTYPE13	R/W	0x00	USB Host Configure Receive Type Endpoint 13	1024
0x1DD	USBRXINTERVAL13	R/W	0x00	USB Host Receive Polling Interval Endpoint 13	1026
0x1E0	USBTXMAXP14	R/W	0x0000	USB Maximum Transmit Data Endpoint 14	986
0x1E2	USBTXCSRL14	R/W	0x00	USB Transmit Control and Status Endpoint 14 Low	997
0x1E3	USBTXCSRH14	R/W	0x00	USB Transmit Control and Status Endpoint 14 High	1002
0x1E4	USBRXMAXP14	R/W	0x0000	USB Maximum Receive Data Endpoint 14	1006
0x1E6	USBRXCSRL14	R/W	0x00	USB Receive Control and Status Endpoint 14 Low	1008
0x1E7	USBRXCSRH14	R/W	0x00	USB Receive Control and Status Endpoint 14 High	1013
0x1E8	USBRXCOUNT14	RO	0x0000	USB Receive Byte Count Endpoint 14	1018
0x1EA	USBTXTYPE14	R/W	0x00	USB Host Transmit Configure Type Endpoint 14	1020
0x1EB	USBTXINTERVAL14	R/W	0x00	USB Host Transmit Interval Endpoint 14	1022
0x1EC	USBRXTYPE14	R/W	0x00	USB Host Configure Receive Type Endpoint 14	1024
0x1ED	USBRXINTERVAL14	R/W	0x00	USB Host Receive Polling Interval Endpoint 14	1026
0x1F0	USBTXMAXP15	R/W	0x0000	USB Maximum Transmit Data Endpoint 15	986
0x1F2	USBTXCSRL15	R/W	0x00	USB Transmit Control and Status Endpoint 15 Low	997
0x1F3	USBTXCSRH15	R/W	0x00	USB Transmit Control and Status Endpoint 15 High	1002
0x1F4	USBRXMAXP15	R/W	0x0000	USB Maximum Receive Data Endpoint 15	1006
0x1F6	USBRXCSRL15	R/W	0x00	USB Receive Control and Status Endpoint 15 Low	1008
0x1F7	USBRXCSRH15	R/W	0x00	USB Receive Control and Status Endpoint 15 High	1013
0x1F8	USBRXCOUNT15	RO	0x0000	USB Receive Byte Count Endpoint 15	1018
0x1FA	USBTXTYPE15	R/W	0x00	USB Host Transmit Configure Type Endpoint 15	1020
0x1FB	USBTXINTERVAL15	R/W	0x00	USB Host Transmit Interval Endpoint 15	1022
0x1FC	USBRXTYPE15	R/W	0x00	USB Host Configure Receive Type Endpoint 15	1024
0x1FD	USBRXINTERVAL15	R/W	0x00	USB Host Receive Polling Interval Endpoint 15	1026
0x304	USBRQPKTCOUNT1	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 1	1028

Table 19-6. Universal Serial Bus (USB) Controller Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x308	USBRQPKTCOUNT2	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 2	1028
0x30C	USBRQPKTCOUNT3	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 3	1028
0x310	USBRQPKTCOUNT4	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 4	1028
0x314	USBRQPKTCOUNT5	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 5	1028
0x318	USBRQPKTCOUNT6	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 6	1028
0x31C	USBRQPKTCOUNT7	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 7	1028
0x320	USBRQPKTCOUNT8	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 8	1028
0x324	USBRQPKTCOUNT9	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 9	1028
0x328	USBRQPKTCOUNT10	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 10	1028
0x32C	USBRQPKTCOUNT11	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 11	1028
0x330	USBRQPKTCOUNT12	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 12	1028
0x334	USBRQPKTCOUNT13	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 13	1028
0x338	USBRQPKTCOUNT14	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 14	1028
0x33C	USBRQPKTCOUNT15	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 15	1028
0x340	USBRXDPKTBUFDIS	R/W	0x0000	USB Receive Double Packet Buffer Disable	1030
0x342	USBTXDPKTBUFDIS	R/W	0x0000	USB Transmit Double Packet Buffer Disable	1032
0x400	USBEPC	R/W	0x0000.0000	USB External Power Control	1034
0x404	USBEPCRIS	RO	0x0000.0000	USB External Power Control Raw Interrupt Status	1037
0x408	USBEPCIM	R/W	0x0000.0000	USB External Power Control Interrupt Mask	1038
0x40C	USBEPCISC	R/W	0x0000.0000	USB External Power Control Interrupt Status and Clear	1039
0x410	USBDRRIS	RO	0x0000.0000	USB Device RESUME Raw Interrupt Status	1040
0x414	USBDRIM	R/W	0x0000.0000	USB Device RESUME Interrupt Mask	1041
0x418	USBDRISC	W1C	0x0000.0000	USB Device RESUME Interrupt Status and Clear	1042
0x41C	USBGPCS	R/W	0x0000.0000	USB General-Purpose Control and Status	1043

Table 19-6. Universal Serial Bus (USB) Controller Register Map (*continued*)

Offset	Name	Type	Reset	Description	See page
0x430	USBVDC	R/W	0x0000.0000	USB VBUS Droop Control	1044
0x434	USBVDCRIS	RO	0x0000.0000	USB VBUS Droop Control Raw Interrupt Status	1045
0x438	USBVDCIM	R/W	0x0000.0000	USB VBUS Droop Control Interrupt Mask	1046
0x43C	USBVDCISC	R/W	0x0000.0000	USB VBUS Droop Control Interrupt Status and Clear	1047
0x444	USBIDVRIS	RO	0x0000.0000	USB ID Valid Detect Raw Interrupt Status	1048
0x448	USBIDVIM	R/W	0x0000.0000	USB ID Valid Detect Interrupt Mask	1049
0x44C	USBIDVISC	R/W1C	0x0000.0000	USB ID Valid Detect Interrupt Status and Clear	1050
0x450	USBDMASEL	R/W	0x0033.2211	USB DMA Select	1051

19.6 Register Descriptions

The LM3S9L97 USB controller has On-The-Go (OTG) capabilities as specified in the **USB0** bit field in the **DC6** register (see page 258).

OTG B / Device

This icon indicates that the register is used in OTG B or Device mode. Some registers are used for both Host and Device mode and may have different bit definitions depending on the mode.

OTG A / Host

This icon indicates that the register is used in OTG A or Host mode. Some registers are used for both Host and Device mode and may have different bit definitions depending on the mode. The USB controller is in OTG B or Device mode upon reset, so the reset values shown for these registers apply to the Device mode definition.

OTG

This icon indicates that the register is used for OTG-specific functions such as ID detection and negotiation. Once OTG negotiation is complete, then the USB controller registers are used according to their Host or Device mode meanings depending on whether the OTG negotiations made the USB controller OTG A (Host) or OTG B (Device).

Register 1: USB Device Functional Address (USBFADDR), offset 0x000

**OTG B /
Device**

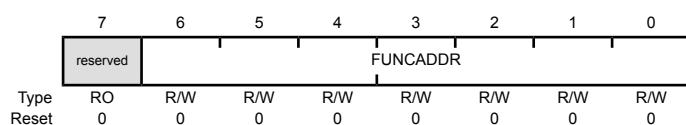
USBFADDR is an 8-bit register that contains the 7-bit address of the Device part of the transaction.

When the USB controller is being used in Device mode (the **HOST** bit in the **USBDEVCTL** register is clear), this register must 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 921 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 SET_ADDRESS.

Register 2: USB Power (USBPOWER), offset 0x001

OTG A / Host

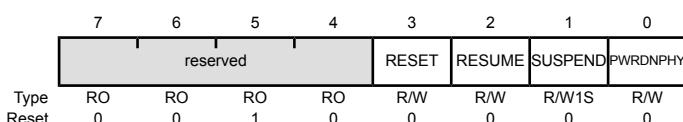
USBPOWER is an 8-bit register used for controlling SUSPEND and RESUME signaling and some basic operational aspects of the USB controller.

OTG B / Device

OTG A / 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	0x2	Software should not rely on the value of 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 Signaling
		Value	Description	
		1	Enables RESET signaling on the bus.	
		0	Ends RESET signaling on the bus.	
2	RESUME	R/W	0	RESUME Signaling
		Value	Description	
		1	Enables RESUME signaling when the Device is in SUSPEND mode.	
		0	Ends RESUME signaling on the bus.	
		This bit must be cleared by software 20 ms after being set.		
1	SUSPEND	R/W1S	0	SUSPEND Mode
		Value	Description	
		1	Enables SUSPEND mode.	
		0	No effect.	

Bit/Field	Name	Type	Reset	Description
0	PWRDNPHY	R/W	0	Power Down PHY
				Value Description
			1	Powers down the internal USB PHY.
			0	No effect.

OTG B / Device Mode

USB Power (USBPOWER)

Base 0x4005.0000

Offset 0x001

Type R/W, reset 0x20

	7	6	5	4	3	2	1	0	
Type	R/W	R/W	RO	RO	RO	R/W	RO	R/W	PWRDNPHY
Reset	0	0	1	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description
7	ISOUP	R/W	0	Isochronous Update
				Value Description
			1	The USB controller waits for an SOF token from the time the TXRDY bit is set in the USBTXCSRLn register before sending the packet. If an IN token is received before an SOF token, then a zero-length data packet is sent.
			0	No effect.
				Note: This bit is only valid for isochronous transfers.
6	SOFTCONN	R/W	0	Soft Connect/Disconnect
				Value Description
			1	The USB D+/D- lines are enabled.
			0	The USB D+/D- lines are tri-stated.
5:4	reserved	RO	0x2	Software should not rely on the value of 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	RO	0	RESET Signaling
				Value Description
			1	RESET signaling is present on the bus.
			0	RESET signaling is not present on the bus.

Bit/Field	Name	Type	Reset	Description						
2	RESUME	R/W	0	<p>RESUME Signaling</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Enables RESUME signaling when the Device is in SUSPEND mode.</td></tr> <tr> <td>0</td><td>Ends RESUME signaling on the bus.</td></tr> </tbody> </table> <p>This bit must be cleared by software 10 ms (a maximum of 15 ms) after being set.</p>	Value	Description	1	Enables RESUME signaling when the Device is in SUSPEND mode.	0	Ends RESUME signaling on the bus.
Value	Description									
1	Enables RESUME signaling when the Device is in SUSPEND mode.									
0	Ends RESUME signaling on the bus.									
1	SUSPEND	RO	0	<p>SUSPEND Mode</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>The USB controller is in SUSPEND mode.</td></tr> <tr> <td>0</td><td>This bit is cleared when software reads the interrupt register or sets the RESUME bit above.</td></tr> </tbody> </table>	Value	Description	1	The USB controller is in SUSPEND mode.	0	This bit is cleared when software reads the interrupt register or sets the RESUME bit above.
Value	Description									
1	The USB controller is in SUSPEND mode.									
0	This bit is cleared when software reads the interrupt register or sets the RESUME bit above.									
0	PWRDNPHY	R/W	0	<p>Power Down PHY</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Powers down the internal USB PHY.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table>	Value	Description	1	Powers down the internal USB PHY.	0	No effect.
Value	Description									
1	Powers down the internal USB PHY.									
0	No effect.									

Register 3: USB Transmit Interrupt Status (USBTXIS), offset 0x002

Important: Use caution when reading this register. Performing a read may change bit status.

**OTG A /
Host**

USBTXIS is a 16-bit read-only register that indicates which interrupts are currently active for endpoint 0 and the transmit endpoints 1–15. The meaning of the EPn bits in this register is based on the mode of the device. The EP1 through EP15 bits always indicate that the USB controller is sending data; however, in Host mode, the bits refer to OUT endpoints; while in Device mode, the bits refer to IN endpoints. The EP0 bit is special in Host and Device modes and indicates that either a control IN or control OUT endpoint has generated an interrupt.

**OTG B /
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 Transmit Interrupt Status (USBTXIS)

Base 0x4005.0000
Offset 0x002
Type RO, reset 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
Reset	RO 0															

Bit/Field	Name	Type	Reset	Description
15	EP15	RO	0	TX Endpoint 15 Interrupt Value Description 0 No interrupt. 1 The Endpoint 15 transmit interrupt is asserted.
14	EP14	RO	0	TX Endpoint 14 Interrupt Same description as EP15.
13	EP13	RO	0	TX Endpoint 13 Interrupt Same description as EP15.
12	EP12	RO	0	TX Endpoint 12 Interrupt Same description as EP15.
11	EP11	RO	0	TX Endpoint 11 Interrupt Same description as EP15.
10	EP10	RO	0	TX Endpoint 10 Interrupt Same description as EP15.
9	EP9	RO	0	TX Endpoint 9 Interrupt Same description as EP15.
8	EP8	RO	0	TX Endpoint 8 Interrupt Same description as EP15.
7	EP7	RO	0	TX Endpoint 7 Interrupt Same description as EP15.

Bit/Field	Name	Type	Reset	Description
6	EP6	RO	0	TX Endpoint 6 Interrupt Same description as EP15.
5	EP5	RO	0	TX Endpoint 5 Interrupt Same description as EP15.
4	EP4	RO	0	TX Endpoint 4 Interrupt Same description as EP15.
3	EP3	RO	0	TX Endpoint 3 Interrupt Same description as EP15.
2	EP2	RO	0	TX Endpoint 2 Interrupt Same description as EP15.
1	EP1	RO	0	TX Endpoint 1 Interrupt Same description as EP15.
0	EP0	RO	0	TX and RX Endpoint 0 Interrupt
				Value Description
			0	No interrupt.
			1	The Endpoint 0 transmit and receive interrupt is asserted.

Register 4: USB Receive Interrupt Status (USBRXIS), offset 0x004

Important: Use caution when reading this register. Performing a read may change bit status.

**OTG A /
Host**

USBRXIS is a 16-bit read-only register that indicates which of the interrupts for receive endpoints 1–15 are currently active.

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.

OTG B /
Device

USB Receive Interrupt Status (USBRXIS)

Base 0x4005.0000

Base 0x1000
Offset 0x004

Type RO, reset 0x0000

Bit/Field	Name	Type	Reset	Description
15	EP15	RO	0	RX Endpoint 15 Interrupt
				Value Description
				0 No interrupt.
				1 The Endpoint 15 receive interrupt is asserted.
14	EP14	RO	0	RX Endpoint 14 Interrupt Same description as EP15.
13	EP13	RO	0	RX Endpoint 13 Interrupt Same description as EP15.
12	EP12	RO	0	RX Endpoint 12 Interrupt Same description as EP15.
11	EP11	RO	0	RX Endpoint 11 Interrupt Same description as EP15.
10	EP10	RO	0	RX Endpoint 10 Interrupt Same description as EP15.
9	EP9	RO	0	RX Endpoint 9 Interrupt Same description as EP15.
8	EP8	RO	0	RX Endpoint 8 Interrupt Same description as EP15.
7	EP7	RO	0	RX Endpoint 7 Interrupt Same description as EP15.
6	EP6	RO	0	RX Endpoint 6 Interrupt Same description as EP15.
5	EP5	RO	0	RX Endpoint 5 Interrupt Same description as EP15.

Bit/Field	Name	Type	Reset	Description
4	EP4	RO	0	RX Endpoint 4 Interrupt Same description as EP15.
3	EP3	RO	0	RX Endpoint 3 Interrupt Same description as EP15.
2	EP2	RO	0	RX Endpoint 2 Interrupt Same description as EP15.
1	EP1	RO	0	RX Endpoint 1 Interrupt Same description as EP15.
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

**OTG A /
Host**

USBTXIE is a 16-bit register that provides interrupt enable bits for the interrupts in the **USBTXIS** register. When a bit is set, the USB interrupt is asserted to the interrupt controller when the corresponding interrupt bit in the **USBTXIS** register is set. When a bit is cleared, the interrupt in the **USBTXIS** register is still set but the USB interrupt to the interrupt controller is not asserted. On reset, all interrupts are enabled.

**OTG B /
Device**

USB Transmit Interrupt Enable (USBTXIE)

Base 0x4005.0000

Offset 0x006

Type R/W, reset 0xFFFF

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0
Reset	R/W 1															

Bit/Field	Name	Type	Reset	Description
15	EP15	R/W	1	TX Endpoint 15 Interrupt Enable
				Value Description
			1	An interrupt is sent to the interrupt controller when the EP15 bit in the USBTXIS register is set.
			0	The EP15 transmit interrupt is suppressed and not sent to the interrupt controller.
14	EP14	R/W	1	TX Endpoint 14 Interrupt Enable Same description as EP15.
13	EP13	R/W	1	TX Endpoint 13 Interrupt Enable Same description as EP15.
12	EP12	R/W	1	TX Endpoint 12 Interrupt Enable Same description as EP15.
11	EP11	R/W	1	TX Endpoint 11 Interrupt Enable Same description as EP15.
10	EP10	R/W	1	TX Endpoint 10 Interrupt Enable Same description as EP15.
9	EP9	R/W	1	TX Endpoint 9 Interrupt Enable Same description as EP15.
8	EP8	R/W	1	TX Endpoint 8 Interrupt Enable Same description as EP15.
7	EP7	R/W	1	TX Endpoint 7 Interrupt Enable Same description as EP15.
6	EP6	R/W	1	TX Endpoint 6 Interrupt Enable Same description as EP15.
5	EP5	R/W	1	TX Endpoint 5 Interrupt Enable Same description as EP15.

Bit/Field	Name	Type	Reset	Description
4	EP4	R/W	1	TX Endpoint 4 Interrupt Enable Same description as EP15.
3	EP3	R/W	1	TX Endpoint 3 Interrupt Enable Same description as EP15.
2	EP2	R/W	1	TX Endpoint 2 Interrupt Enable Same description as EP15.
1	EP1	R/W	1	TX Endpoint 1 Interrupt Enable Same description as EP15.
0	EP0	R/W	1	TX and RX Endpoint 0 Interrupt Enable
				Value Description
				1 An interrupt is sent to the interrupt controller when the EP0 bit in the USBTXIS register is set.
				0 The EP0 transmit and receive interrupt is suppressed and not sent to the interrupt controller.

Register 6: USB Receive Interrupt Enable (USBRXIE), offset 0x008

**OTG A /
Host**

USBRXIE is a 16-bit register that provides interrupt enable bits for the interrupts in the **USBRXIS** register. When a bit is set, the USB interrupt is asserted to the interrupt controller when the corresponding interrupt bit in the **USBRXIS** register is set. When a bit is cleared, the interrupt in the **USBRXIS** register is still set but the USB interrupt to the interrupt controller is not asserted. On reset, all interrupts are enabled.

**OTG B /
Device**

USB Receive Interrupt Enable (USBRXIE)

Base 0x4005.0000

Offset 0x008

Type R/W, reset 0xFFFF

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	reserved
Reset	R/W 1	RO 0														

Bit/Field	Name	Type	Reset	Description
15	EP15	R/W	1	RX Endpoint 15 Interrupt Enable
				Value Description
			1	An interrupt is sent to the interrupt controller when the EP15 bit in the USBRXIS register is set.
			0	The EP15 receive interrupt is suppressed and not sent to the interrupt controller.
14	EP14	R/W	1	RX Endpoint 14 Interrupt Enable Same description as EP15.
13	EP13	R/W	1	RX Endpoint 13 Interrupt Enable Same description as EP15.
12	EP12	R/W	1	RX Endpoint 12 Interrupt Enable Same description as EP15.
11	EP11	R/W	1	RX Endpoint 11 Interrupt Enable Same description as EP15.
10	EP10	R/W	1	RX Endpoint 10 Interrupt Enable Same description as EP15.
9	EP9	R/W	1	RX Endpoint 9 Interrupt Enable Same description as EP15.
8	EP8	R/W	1	RX Endpoint 8 Interrupt Enable Same description as EP15.
7	EP7	R/W	1	RX Endpoint 7 Interrupt Enable Same description as EP15.
6	EP6	R/W	1	RX Endpoint 6 Interrupt Enable Same description as EP15.
5	EP5	R/W	1	RX Endpoint 5 Interrupt Enable Same description as EP15.

Bit/Field	Name	Type	Reset	Description
4	EP4	R/W	1	RX Endpoint 4 Interrupt Enable Same description as EP15.
3	EP3	R/W	1	RX Endpoint 3 Interrupt Enable Same description as EP15.
2	EP2	R/W	1	RX Endpoint 2 Interrupt Enable Same description as EP15.
1	EP1	R/W	1	RX Endpoint 1 Interrupt Enable Same description as EP15.
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

Important: Use caution when reading this register. Performing a read may change bit status.

OTG A /
Host

OTG B /
Device

OTG A / Host Mode

USB General Interrupt Status (USBIS)

Base 0x4005.0000
Offset 0x00A
Type RO, reset 0x00

	7	6	5	4	3	2	1	0
Type	VBUSERR	SESREQ	DISCON	CONN	SOF	BABBLE	RESUME	reserved
Reset	RO 0							

Bit/Field	Name	Type	Reset	Description
7	VBUSERR	RO	0	VBUS Error Value Description 1 VBUS has dropped below the VBUS Valid threshold during a session. 0 No interrupt.
6	SESREQ	RO	0	SESSION REQUEST Value Description 1 SESSION REQUEST signaling has been detected. 0 No interrupt.
5	DISCON	RO	0	Session Disconnect Value Description 1 A Device disconnect has been detected. 0 No interrupt.
4	CONN	RO	0	Session Connect Value Description 1 A Device connection has been detected. 0 No interrupt.

Bit/Field	Name	Type	Reset	Description
3	SOF	RO	0	<p>Start of Frame</p> <p>Value Description</p> <p>1 A new frame has started.</p> <p>0 No interrupt.</p>
2	BABBLE	RO	0	<p>Babble Detected</p> <p>Value Description</p> <p>1 Babble has been detected. This interrupt is active only after the first SOF has been sent.</p> <p>0 No interrupt.</p>
1	RESUME	RO	0	<p>RESUME Signaling Detected</p> <p>Value Description</p> <p>1 RESUME signaling has been detected on the bus while the USB controller is in SUSPEND mode.</p> <p>0 No interrupt.</p> <p>This interrupt can only be used if the USB controller's system clock is enabled. If the user disables the clock programming, the USBDRRIS, USBDRIM, and USBDRISC registers should be used.</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.

OTG B / Device Mode

USB General Interrupt Status (USBIS)

Base 0x4005.0000

Offset 0x0A

Type RO, reset 0x00

	7	6	5	4	3	2	1	0
Type	reserved	DISCON	reserved	SOF	RESET	RESUME	SUSPEND	
Reset	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0	RO 0

Bit/Field	Name	Type	Reset	Description
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	DISCON	RO	0	<p>Session Disconnect</p> <p>Value Description</p> <p>1 The device has been disconnected from the host.</p> <p>0 No interrupt.</p>

Bit/Field	Name	Type	Reset	Description
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	Start of Frame Value Description 1 A new frame has started. 0 No interrupt.
2	RESET	RO	0	RESET Signaling Detected Value Description 1 RESET signaling has been detected on the bus. 0 No interrupt.
1	RESUME	RO	0	RESUME Signaling Detected Value Description 1 RESUME signaling has been detected on the bus while the USB controller is in SUSPEND mode. 0 No interrupt. This interrupt can only be used if the USB controller's system clock is enabled. If the user disables the clock programming, the USBDRRIS , USBDRIM , and USBDRISC registers should be used.
0	SUSPEND	RO	0	SUSPEND Signaling Detected Value Description 1 SUSPEND signaling has been detected on the bus. 0 No interrupt.

Register 8: USB Interrupt Enable (USBIE), offset 0x00B

OTG A / Host

USBIE is an 8-bit register that provides interrupt enable bits for each of the interrupts in **USBIS**. At reset interrupts 1 and 2 are enabled in Device mode.

OTG B / Device

OTG A / Host Mode

USB Interrupt Enable (USBIE)

Base 0x4005.0000
Offset 0x00B
Type R/W, reset 0x06

	7	6	5	4	3	2	1	0
Type	R/W	RO						
Reset	0	0	0	0	0	1	1	0

Bit/Field	Name	Type	Reset	Description
7	VBUSERR	R/W	0	Enable VBUS Error Interrupt
				Value Description
			1	An interrupt is sent to the interrupt controller when the VBUSERR bit in the USBIS register is set.
			0	The VBUSERR interrupt is suppressed and not sent to the interrupt controller.
6	SESREQ	R/W	0	Enable Session Request
				Value Description
			1	An interrupt is sent to the interrupt controller when the SESREQ bit in the USBIS register is set.
			0	The SESREQ interrupt is suppressed and not sent to the interrupt controller.
5	DISCON	R/W	0	Enable Disconnect Interrupt
				Value Description
			1	An interrupt is sent to the interrupt controller when the DISCON bit in the USBIS register is set.
			0	The DISCON interrupt is suppressed and not sent to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
4	CONN	R/W	0	Enable Connect Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the CONN bit in the USBIS register is set. 0 The CONN interrupt is suppressed and not sent to the interrupt controller.
3	SOF	R/W	0	Enable Start-of-Frame Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the SOF bit in the USBIS register is set. 0 The SOF interrupt is suppressed and not sent to the interrupt controller.
2	BABBLE	R/W	1	Enable Babble Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the BABBLE bit in the USBIS register is set. 0 The BABBLE interrupt is suppressed and not sent to the interrupt controller.
1	RESUME	R/W	1	Enable RESUME Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the RESUME bit in the USBIS register is set. 0 The RESUME 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.

OTG 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
	reserved		DISCON	reserved	SOF	RESET	RESUME	SUSPEND
Type	RO	RO	R/W	RO	R/W	R/W	R/W	R/W

Bit/Field	Name	Type	Reset	Description
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	DISCON	R/W	0	Enable Disconnect Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the DISCON bit in the USBIS register is set. 0 The DISCON interrupt is suppressed and not sent to the interrupt controller.
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	R/W	0	Enable Start-of-Frame Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the SOF bit in the USBIS register is set. 0 The SOF interrupt is suppressed and not sent to the interrupt controller.
2	RESET	R/W	1	Enable RESET Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the RESET bit in the USBIS register is set. 0 The RESET interrupt is suppressed and not sent to the interrupt controller.
1	RESUME	R/W	1	Enable RESUME Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the RESUME bit in the USBIS register is set. 0 The RESUME interrupt is suppressed and not sent to the interrupt controller.
0	SUSPEND	R/W	0	Enable SUSPEND Interrupt Value Description 1 An interrupt is sent to the interrupt controller when the SUSPEND bit in the USBIS register is set. 0 The SUSPEND interrupt is suppressed and not sent to the interrupt controller.

Register 9: USB Frame Value (USBFRAME), offset 0x00C

**OTG A /
Host**

USBFRAME is a 16-bit read-only register that holds the last received frame number.

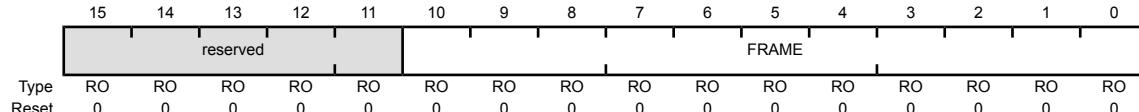
USB Frame Value (USBFRAME)

Base 0x4005.0000

Offset 0x00C

Type RO, reset 0x0000

**OTG B /
Device**



Bit/Field	Name	Type	Reset	Description
15:11	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.
10:0	FRAME	RO	0x000	Frame Number

Register 10: USB Endpoint Index (USBEPIDX), offset 0x00E

**OTG A /
Host**

Each endpoint's buffer can be accessed by configuring a FIFO size and starting address. The **USBEPIDX** 8-bit register is used with the **USBTXFIFOSZ**, **USBRXFIFOSZ**, **USBTXFIFOADD**, and **USBRXFIFOADD** registers.

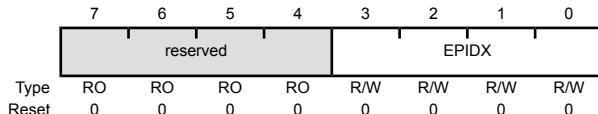
**OTG B /
Device**

USB Endpoint Index (USBEPIDX)

Base 0x4005.0000

Offset 0x00E

Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:4	reserved	RO	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	EPIDX	R/W	0x0	Endpoint Index This bit field configures which endpoint is accessed when reading or writing to one of the USB controller's indexed registers. A value of 0x0 corresponds to Endpoint 0 and a value of 0xF corresponds to Endpoint 15.

Register 11: USB Test Mode (USBTEST), offset 0x00F

OTG A / Host

USBTEST 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. This register is not used in normal operation.

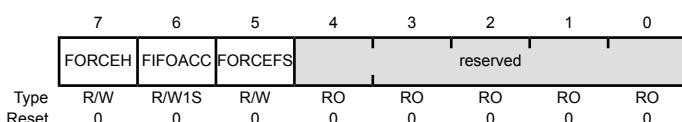
Note: Only one of these bits should be set at any time.

OTG B / Device

OTG A / 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	Force Host Mode

Value	Description
1	Forces the USB controller to enter Host mode when the SESSION bit is set, regardless of whether the USB controller is connected to any peripheral. The state of the USB0DP and USB0DM signals is ignored. The USB controller then remains in Host mode until the SESSION bit is cleared, even if a Device is disconnected. If the FORCEH bit remains set, the USB controller re-enters Host mode the next time the SESSION bit is set.
0	No effect.

While in this mode, status of the bus connection may be read using the DEV bit of the **USBDEVCTL** register. The operating speed is determined from the FORCEFS bit.

6	FIFOACC	R/W1S	0	FIFO Access
---	---------	-------	---	-------------

Value	Description
1	Transfers the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO.
0	No effect.

This bit is cleared automatically.

5	FORCEFS	R/W	0	Force Full-Speed Mode
---	---------	-----	---	-----------------------

Value	Description
1	Forces the USB controller into Full-Speed mode upon receiving a USB RESET.
0	The USB controller operates at Low Speed.

Bit/Field	Name	Type	Reset	Description
4:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

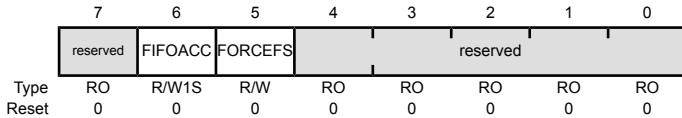
OTG B / Device Mode

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 Value Description 1 Transfers the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO. 0 No effect. This bit is cleared automatically.
5	FORCEFS	R/W	0	Force Full-Speed Mode Value Description 1 Forces the USB controller into Full-Speed mode upon receiving a USB RESET. 0 The USB controller operates at Low Speed.
4:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

- Register 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**
- Register 16: USB FIFO Endpoint 4 (USBFIFO4), offset 0x030**
- Register 17: USB FIFO Endpoint 5 (USBFIFO5), offset 0x034**
- Register 18: USB FIFO Endpoint 6 (USBFIFO6), offset 0x038**
- Register 19: USB FIFO Endpoint 7 (USBFIFO7), offset 0x03C**
- Register 20: USB FIFO Endpoint 8 (USBFIFO8), offset 0x040**
- Register 21: USB FIFO Endpoint 9 (USBFIFO9), offset 0x044**
- Register 22: USB FIFO Endpoint 10 (USBFIFO10), offset 0x048**
- Register 23: USB FIFO Endpoint 11 (USBFIFO11), offset 0x04C**
- Register 24: USB FIFO Endpoint 12 (USBFIFO12), offset 0x050**
- Register 25: USB FIFO Endpoint 13 (USBFIFO13), offset 0x054**
- Register 26: USB FIFO Endpoint 14 (USBFIFO14), offset 0x058**
- Register 27: USB FIFO Endpoint 15 (USBFIFO15), offset 0x05C**

Important: Use caution when reading this register. Performing a read may change bit status.

**OTG A /
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.

**OTG B /
Device**

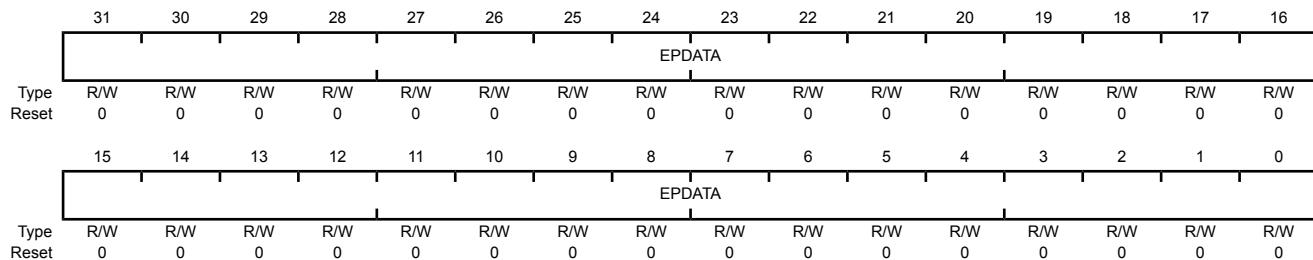
Transfers to and from FIFOs may be 8-bit, 16-bit or 32-bit as required, and any combination of accesses is allowed provided the data accessed is contiguous. All transfers associated with one packet must be of the same width so that the data is consistently byte-, halfword- or word-aligned. However, the 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 (see the section called “Single-Packet Buffering” on page 919). Burst writing of multiple packets is not supported as flags must be set after each packet is written.

Following a STALL response or a transmit error on endpoint 1–15, 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	0x0000.0000	Endpoint Data Writing to this register loads the data into the Transmit FIFO and reading unloads data from the Receive FIFO.

Register 28: USB Device Control (USBDEVCTL), offset 0x060

**OTG A /
Host**

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. In addition, in Host mode, **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.

USB Device Control (USBDEVCTL)

Base 0x4005.0000
Offset 0x060
Type R/W, reset 0x80

	7	6	5	4	3	2	1	0
Type	DEV	FSDEV	LSDEV	VBUS	HOST	HOSTREQ	SESSION	
Reset	RO 1	RO 0	RO 0	RO 0	RO 0	RO 0	R/W 0	R/W 0

Bit/Field	Name	Type	Reset	Description
7	DEV	RO	1	Device Mode Value Description 0 The USB controller is operating on the OTG A side of the cable. 1 The USB controller is operating on the OTG B side of the cable. Note: This value is only valid while a session is in progress.
6	FSDEV	RO	0	Full-Speed Device Detected Value Description 0 A full-speed Device has not been detected on the port. 1 A full-speed Device has been detected on the port.
5	LSDEV	RO	0	Low-Speed Device Detected Value Description 0 A low-speed Device has not been detected on the port. 1 A low-speed Device has been detected on the port.
4:3	VBUS	RO	0x0	VBUS Level 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 4.75 V. 0x3 Above VBUSValid VBUS is detected as above 4.75 V.

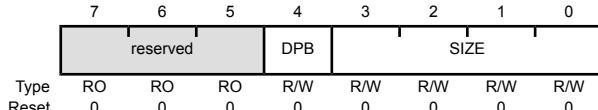
Bit/Field	Name	Type	Reset	Description												
2	HOST	RO	0	<p>Host Mode</p> <table> <tr> <td>Value</td><td>Description</td></tr> <tr> <td>0</td><td>The USB controller is acting as a Device.</td></tr> <tr> <td>1</td><td>The USB controller is acting as a Host.</td></tr> </table> <p>Note: This value is only valid while a session is in progress.</p>	Value	Description	0	The USB controller is acting as a Device.	1	The USB controller is acting as a Host.						
Value	Description															
0	The USB controller is acting as a Device.															
1	The USB controller is acting as a Host.															
1	HOSTREQ	R/W	0	<p>Host Request</p> <table> <tr> <td>Value</td><td>Description</td></tr> <tr> <td>0</td><td>No effect.</td></tr> <tr> <td>1</td><td>Initiates the Host Negotiation when SUSPEND mode is entered.</td></tr> </table> <p>This bit is cleared when Host Negotiation is completed.</p>	Value	Description	0	No effect.	1	Initiates the Host Negotiation when SUSPEND mode is entered.						
Value	Description															
0	No effect.															
1	Initiates the Host Negotiation when SUSPEND mode is entered.															
0	SESSION	R/W	0	<p>Session Start/End</p> <p><i>When operating as an OTG A device:</i></p> <table> <tr> <td>Value</td><td>Description</td></tr> <tr> <td>0</td><td>When cleared by software, this bit ends a session.</td></tr> <tr> <td>1</td><td>When set by software, this bit starts a session.</td></tr> </table> <p><i>When operating as an OTG B device:</i></p> <table> <tr> <td>Value</td><td>Description</td></tr> <tr> <td>0</td><td>The USB controller has ended a session. When the USB controller is in SUSPEND mode, this bit may be cleared by software to perform a software disconnect.</td></tr> <tr> <td>1</td><td>The USB controller has started a session. When set by software, the Session Request Protocol is initiated.</td></tr> </table> <p>Note: Clearing this bit when the USB controller is not suspended results in undefined behavior.</p>	Value	Description	0	When cleared by software, this bit ends a session.	1	When set by software, this bit starts a session.	Value	Description	0	The USB controller has ended a session. When the USB controller is in SUSPEND mode, this bit may be cleared by software to perform a software disconnect.	1	The USB controller has started a session. When set by software, the Session Request Protocol is initiated.
Value	Description															
0	When cleared by software, this bit ends a session.															
1	When set by software, this bit starts a session.															
Value	Description															
0	The USB controller has ended a session. When the USB controller is in SUSPEND mode, this bit may be cleared by software to perform a software disconnect.															
1	The USB controller has started a session. When set by software, the Session Request Protocol is initiated.															

Register 29: USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ), offset 0x062**Register 30: USB Receive Dynamic FIFO Sizing (USBRXFIFOSZ), offset 0x063****OTG A /
Host**

These 8-bit registers allow the selected TX/RX endpoint FIFOs to be dynamically sized. **USBEPIIDX** is used to configure each transmit endpoint's FIFO size.

**OTG B /
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	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	DPB	R/W	0	Double Packet Buffer Support Value Description 0 Only single-packet buffering is supported. 1 Double-packet buffering is supported.
3:0	SIZE	R/W	0x0	Max Packet Size Maximum packet size to be allowed. If DPB = 0, the FIFO also is this size; if DPB = 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

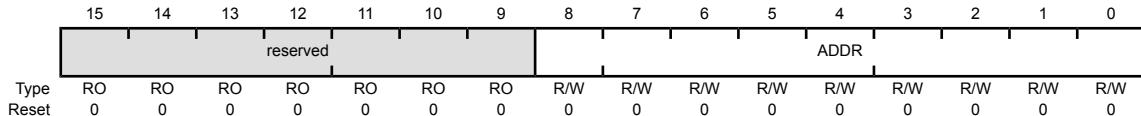
Register 31: USB Transmit FIFO Start Address (USBTXFIFOADD), offset 0x064**Register 32: USB Receive FIFO Start Address (USBRXFIFOADD), offset 0x066****OTG A /
Host****USBTXFIFOADD** and **USBRXFIFOADD** are 16-bit registers that control the start address of the selected transmit and receive endpoint FIFOs.

USB Transmit FIFO Start Address (USBTXFIFOADD)

Base 0x4005.0000

Offset 0x064

Type R/W, reset 0x0000

**OTG B /
Device**

Bit/Field	Name	Type	Reset	Description
15:9	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.
8:0	ADDR	R/W	0x00	Transmit/Receive Start Address Start address of the endpoint FIFO.

Value	Start Address
0x0	0
0x1	8
0x2	16
0x3	24
0x4	32
0x5	40
0x6	48
0x7	56
0x8	64
...	...
0x1FF	4095

Register 33: USB Connect Timing (USBCONTIM), offset 0x07A

**OTG A /
Host**

This 8-bit configuration register specifies connection and negotiation delays.

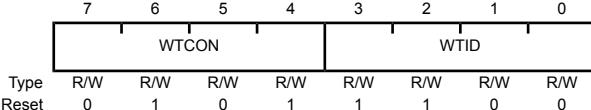
USB Connect Timing (USBCONTIM)

Base 0x4005.0000

Offset 0x07A

Type R/W, reset 0x5C

**OTG B /
Device**



Bit/Field	Name	Type	Reset	Description
7:4	WTCON	R/W	0x5	Connect Wait This field configures the wait required to allow for the user's connect/disconnect filter, in units of 533.3 ns. The default corresponds to 2.667 µs.
3:0	WTID	R/W	0xC	Wait ID This field configures the delay required from the enable of the ID detection to when the ID value is valid, in units of 4.369 ms. The default corresponds to 52.43 ms.

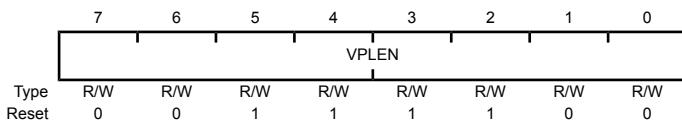
Register 34: USB OTG VBUS Pulse Timing (USBVPLEN), offset 0x07B

OTG

This 8-bit configuration register specifies 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 This field configures the duration of the VBUS pulsing charge in units of 546.1 µs. The default corresponds to 32.77 ms.

Register 35: USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF), offset 0x07D

**OTG A /
Host**

This 8-bit configuration register specifies the minimum time gap allowed between the start of the last transaction and the EOF for full-speed transactions.

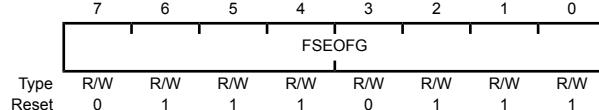
USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF)

Base 0x4005.0000

Offset 0x07D

Type R/W, reset 0x77

**OTG B /
Device**



Bit/Field	Name	Type	Reset	Description
7:0	FSEOFG	R/W	0x77	<p>Full-Speed End-of-Frame Gap</p> <p>This field is used during full-speed transactions to configure the gap between the last transaction and the End-of-Frame (EOF), in units of 533.3 ns. The default corresponds to 63.46 µs.</p>

Register 36: USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF), offset 0x07E

**OTG A /
Host**

This 8-bit configuration register specifies the minimum time gap that is to be allowed between the start of the last transaction and the EOF for low-speed transactions.

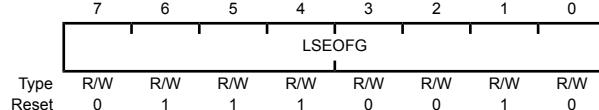
USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF)

**OTG B /
Device**

Base 0x4005.0000

Offset 0x07E

Type R/W, reset 0x72



Bit/Field	Name	Type	Reset	Description
7:0	LSEOFG	R/W	0x72	<p>Low-Speed End-of-Frame Gap</p> <p>This field is used during low-speed transactions to set the gap between the last transaction and the End-of-Frame (EOF), in units of 1.067 µs. The default corresponds to 121.6 µs.</p>

**Register 37: USB Transmit Functional Address Endpoint 0
(USBTXFUNCADDR0), offset 0x080**

**Register 38: USB Transmit Functional Address Endpoint 1
(USBTXFUNCADDR1), offset 0x088**

**Register 39: USB Transmit Functional Address Endpoint 2
(USBTXFUNCADDR2), offset 0x090**

**Register 40: USB Transmit Functional Address Endpoint 3
(USBTXFUNCADDR3), offset 0x098**

**Register 41: USB Transmit Functional Address Endpoint 4
(USBTXFUNCADDR4), offset 0x0A0**

**Register 42: USB Transmit Functional Address Endpoint 5
(USBTXFUNCADDR5), offset 0x0A8**

**Register 43: USB Transmit Functional Address Endpoint 6
(USBTXFUNCADDR6), offset 0x0B0**

**Register 44: USB Transmit Functional Address Endpoint 7
(USBTXFUNCADDR7), offset 0x0B8**

**Register 45: USB Transmit Functional Address Endpoint 8
(USBTXFUNCADDR8), offset 0x0C0**

**Register 46: USB Transmit Functional Address Endpoint 9
(USBTXFUNCADDR9), offset 0x0C8**

**Register 47: USB Transmit Functional Address Endpoint 10
(USBTXFUNCADDR10), offset 0x0D0**

**Register 48: USB Transmit Functional Address Endpoint 11
(USBTXFUNCADDR11), offset 0x0D8**

**Register 49: USB Transmit Functional Address Endpoint 12
(USBTXFUNCADDR12), offset 0x0E0**

**Register 50: USB Transmit Functional Address Endpoint 13
(USBTXFUNCADDR13), offset 0x0E8**

**Register 51: USB Transmit Functional Address Endpoint 14
(USBTXFUNCADDR14), offset 0x0F0**

**Register 52: USB Transmit Functional Address Endpoint 15
(USBTXFUNCADDR15), offset 0x0F8**

OTG A /
Host

USBTXFUNCADDRn is an 8-bit read/write register that records the address of the target function to be accessed through the associated endpoint (EPn). **USBTXFUNCADDRn** must be defined for each transmit endpoint that is used.

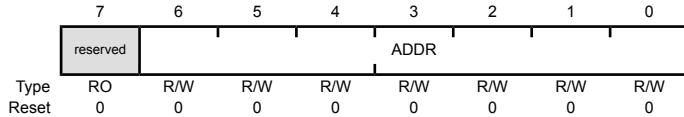
Note: **USBTXFUNCADDR0** is used for both receive and transmit for endpoint 0.

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 Specifies the USB bus address for the target Device.

Register 53: USB Transmit Hub Address Endpoint 0 (USBTXHUBADDR0), offset 0x082

Register 54: USB Transmit Hub Address Endpoint 1 (USBTXHUBADDR1), offset 0x08A

Register 55: USB Transmit Hub Address Endpoint 2 (USBTXHUBADDR2), offset 0x092

Register 56: USB Transmit Hub Address Endpoint 3 (USBTXHUBADDR3), offset 0x09A

Register 57: USB Transmit Hub Address Endpoint 4 (USBTXHUBADDR4), offset 0x0A2

Register 58: USB Transmit Hub Address Endpoint 5 (USBTXHUBADDR5), offset 0x0AA

Register 59: USB Transmit Hub Address Endpoint 6 (USBTXHUBADDR6), offset 0x0B2

Register 60: USB Transmit Hub Address Endpoint 7 (USBTXHUBADDR7), offset 0x0BA

Register 61: USB Transmit Hub Address Endpoint 8 (USBTXHUBADDR8), offset 0x0C2

Register 62: USB Transmit Hub Address Endpoint 9 (USBTXHUBADDR9), offset 0x0CA

Register 63: USB Transmit Hub Address Endpoint 10 (USBTXHUBADDR10), offset 0x0D2

Register 64: USB Transmit Hub Address Endpoint 11 (USBTXHUBADDR11), offset 0x0DA

Register 65: USB Transmit Hub Address Endpoint 12 (USBTXHUBADDR12), offset 0x0E2

Register 66: USB Transmit Hub Address Endpoint 13 (USBTXHUBADDR13), offset 0x0EA

Register 67: USB Transmit Hub Address Endpoint 14 (USBTXHUBADDR14), offset 0x0F2

Register 68: USB Transmit Hub Address Endpoint 15 (USBTXHUBADDR15), offset 0x0FA

OTG A /
Host

USBTXHUBADDRn is an 8-bit read/write register that, like USBTXHUBPORTn, only must be written when a USB Device is connected to transmit endpoint EPn via a USB 2.0 hub. This register records the address of the USB 2.0 hub through which the target associated with the endpoint is accessed.

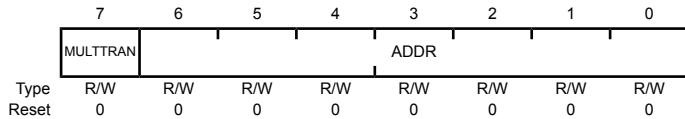
Note: USBTXHUBADDR0 is used for both receive and transmit for endpoint 0.

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
---	----------	-----	---	----------------------

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

0	Clear to indicate that the hub has a single transaction translator.
1	Set to indicate that the hub has multiple transaction translators.

6:0	ADDR	R/W	0x00	Hub Address
-----	------	-----	------	-------------

This field specifies the USB bus address for the USB 2.0 hub.

Register 69: USB Transmit Hub Port Endpoint 0 (USBTXHUBPORT0), offset 0x083

Register 70: USB Transmit Hub Port Endpoint 1 (USBTXHUBPORT1), offset 0x08B

Register 71: USB Transmit Hub Port Endpoint 2 (USBTXHUBPORT2), offset 0x093

Register 72: USB Transmit Hub Port Endpoint 3 (USBTXHUBPORT3), offset 0x09B

Register 73: USB Transmit Hub Port Endpoint 4 (USBTXHUBPORT4), offset 0x0A3

Register 74: USB Transmit Hub Port Endpoint 5 (USBTXHUBPORT5), offset 0x0AB

Register 75: USB Transmit Hub Port Endpoint 6 (USBTXHUBPORT6), offset 0x0B3

Register 76: USB Transmit Hub Port Endpoint 7 (USBTXHUBPORT7), offset 0x0BB

Register 77: USB Transmit Hub Port Endpoint 8 (USBTXHUBPORT8), offset 0x0C3

Register 78: USB Transmit Hub Port Endpoint 9 (USBTXHUBPORT9), offset 0x0CB

Register 79: USB Transmit Hub Port Endpoint 10 (USBTXHUBPORT10), offset 0x0D3

Register 80: USB Transmit Hub Port Endpoint 11 (USBTXHUBPORT11), offset 0x0DB

Register 81: USB Transmit Hub Port Endpoint 12 (USBTXHUBPORT12), offset 0x0E3

Register 82: USB Transmit Hub Port Endpoint 13 (USBTXHUBPORT13), offset 0x0EB

Register 83: USB Transmit Hub Port Endpoint 14 (USBTXHUBPORT14), offset 0x0F3

Register 84: USB Transmit Hub Port Endpoint 15 (USBTXHUBPORT15), offset 0x0FB

OTG A /
Host

USBTXHUBPORTn is an 8-bit read/write register that, like **USBTXHUBADDRn**, only must 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 the USB 2.0 hub through which the target associated with the endpoint is accessed.

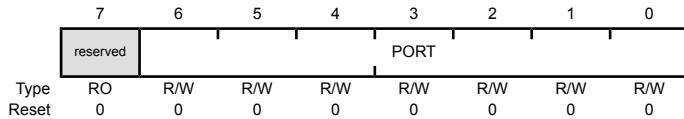
Note: **USBTXHUBPORT0** is used for both receive and transmit for endpoint 0.

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 This field specifies the USB hub port number.

**Register 85: USB Receive Functional Address Endpoint 1
(USBRXFUNCADDR1), offset 0x08C**

**Register 86: USB Receive Functional Address Endpoint 2
(USBRXFUNCADDR2), offset 0x094**

**Register 87: USB Receive Functional Address Endpoint 3
(USBRXFUNCADDR3), offset 0x09C**

**Register 88: USB Receive Functional Address Endpoint 4
(USBRXFUNCADDR4), offset 0xA4**

**Register 89: USB Receive Functional Address Endpoint 5
(USBRXFUNCADDR5), offset 0x0AC**

**Register 90: USB Receive Functional Address Endpoint 6
(USBRXFUNCADDR6), offset 0x0B4**

**Register 91: USB Receive Functional Address Endpoint 7
(USBRXFUNCADDR7), offset 0x0BC**

**Register 92: USB Receive Functional Address Endpoint 8
(USBRXFUNCADDR8), offset 0x0C4**

**Register 93: USB Receive Functional Address Endpoint 9
(USBRXFUNCADDR9), offset 0x0CC**

**Register 94: USB Receive Functional Address Endpoint 10
(USBRXFUNCADDR10), offset 0x0D4**

**Register 95: USB Receive Functional Address Endpoint 11
(USBRXFUNCADDR11), offset 0x0DC**

**Register 96: USB Receive Functional Address Endpoint 12
(USBRXFUNCADDR12), offset 0x0E4**

**Register 97: USB Receive Functional Address Endpoint 13
(USBRXFUNCADDR13), offset 0x0EC**

**Register 98: USB Receive Functional Address Endpoint 14
(USBRXFUNCADDR14), offset 0x0F4**

**Register 99: USB Receive Functional Address Endpoint 15
(USBRXFUNCADDR15), offset 0x0FC**

OTG A / Host
USBRXFUNCADDRn is an 8-bit read/write register that records the address of the target function accessed through the associated endpoint (EPn). **USBRXFUNCADDRn** must be defined for each receive endpoint that is used.

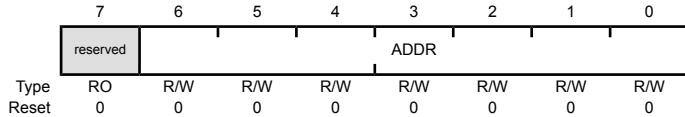
Note: **USBTXFUNCADDR0** is used for both receive and transmit for endpoint 0.

USB Receive Functional Address Endpoint 1 (USBRXFUNCADDR1)

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 This field specifies the USB bus address for the target Device.

Register 100: USB Receive Hub Address Endpoint 1 (USBRXHUBADDR1), offset 0x08E

Register 101: USB Receive Hub Address Endpoint 2 (USBRXHUBADDR2), offset 0x096

Register 102: USB Receive Hub Address Endpoint 3 (USBRXHUBADDR3), offset 0x09E

Register 103: USB Receive Hub Address Endpoint 4 (USBRXHUBADDR4), offset 0x0A6

Register 104: USB Receive Hub Address Endpoint 5 (USBRXHUBADDR5), offset 0x0AE

Register 105: USB Receive Hub Address Endpoint 6 (USBRXHUBADDR6), offset 0x0B6

Register 106: USB Receive Hub Address Endpoint 7 (USBRXHUBADDR7), offset 0x0BE

Register 107: USB Receive Hub Address Endpoint 8 (USBRXHUBADDR8), offset 0x0C6

Register 108: USB Receive Hub Address Endpoint 9 (USBRXHUBADDR9), offset 0x0CE

Register 109: USB Receive Hub Address Endpoint 10 (USBRXHUBADDR10), offset 0x0D6

Register 110: USB Receive Hub Address Endpoint 11 (USBRXHUBADDR11), offset 0x0DE

Register 111: USB Receive Hub Address Endpoint 12 (USBRXHUBADDR12), offset 0x0E6

Register 112: USB Receive Hub Address Endpoint 13 (USBRXHUBADDR13), offset 0x0EE

Register 113: USB Receive Hub Address Endpoint 14 (USBRXHUBADDR14), offset 0x0F6

Register 114: USB Receive Hub Address Endpoint 15 (USBRXHUBADDR15), offset 0x0FE

OTG A /
Host

USBRXHUBADDRn is an 8-bit read/write register that, like **USBRXHUBPORTn**, only must 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 the USB 2.0 hub through which the target associated with the endpoint is accessed.

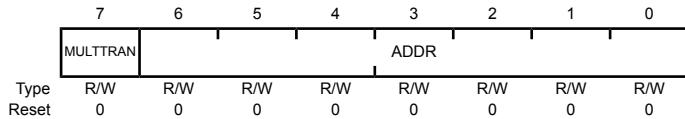
Note: **USBTXHUBADDR0** is used for both receive and transmit for endpoint 0.

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
				Value Description
			0	Clear to indicate that the hub has a single transaction translator.
			1	Set to indicate that the hub has multiple transaction translators.
6:0	ADDR	R/W	0x00	Hub Address This field specifies the USB bus address for the USB 2.0 hub.

Register 115: USB Receive Hub Port Endpoint 1 (USBRXHUBPORT1), offset 0x08F

Register 116: USB Receive Hub Port Endpoint 2 (USBRXHUBPORT2), offset 0x097

Register 117: USB Receive Hub Port Endpoint 3 (USBRXHUBPORT3), offset 0x09F

Register 118: USB Receive Hub Port Endpoint 4 (USBRXHUBPORT4), offset 0x0A7

Register 119: USB Receive Hub Port Endpoint 5 (USBRXHUBPORT5), offset 0x0AF

Register 120: USB Receive Hub Port Endpoint 6 (USBRXHUBPORT6), offset 0x0B7

Register 121: USB Receive Hub Port Endpoint 7 (USBRXHUBPORT7), offset 0x0BF

Register 122: USB Receive Hub Port Endpoint 8 (USBRXHUBPORT8), offset 0x0C7

Register 123: USB Receive Hub Port Endpoint 9 (USBRXHUBPORT9), offset 0x0CF

Register 124: USB Receive Hub Port Endpoint 10 (USBRXHUBPORT10), offset 0x0D7

Register 125: USB Receive Hub Port Endpoint 11 (USBRXHUBPORT11), offset 0x0DF

Register 126: USB Receive Hub Port Endpoint 12 (USBRXHUBPORT12), offset 0x0E7

Register 127: USB Receive Hub Port Endpoint 13 (USBRXHUBPORT13), offset 0x0EF

Register 128: USB Receive Hub Port Endpoint 14 (USBRXHUBPORT14), offset 0x0F7

Register 129: USB Receive Hub Port Endpoint 15 (USBRXHUBPORT15), offset 0x0FF

OTG A /
Host

USBRXHUBPORTn is an 8-bit read/write register that, like **USBRXHUBADDRn**, only must 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 port of the USB 2.0 hub through which the target associated with the endpoint is accessed.

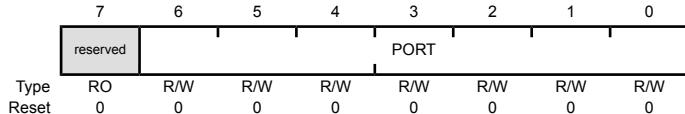
Note: **USBTXHUBPORT0** is used for both receive and transmit for endpoint 0.

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 This field specifies the USB hub port number.

Register 130: USB Maximum Transmit Data Endpoint 1 (USBTXMAXP1), offset 0x110

Register 131: USB Maximum Transmit Data Endpoint 2 (USBTXMAXP2), offset 0x120

Register 132: USB Maximum Transmit Data Endpoint 3 (USBTXMAXP3), offset 0x130

Register 133: USB Maximum Transmit Data Endpoint 4 (USBTXMAXP4), offset 0x140

Register 134: USB Maximum Transmit Data Endpoint 5 (USBTXMAXP5), offset 0x150

Register 135: USB Maximum Transmit Data Endpoint 6 (USBTXMAXP6), offset 0x160

Register 136: USB Maximum Transmit Data Endpoint 7 (USBTXMAXP7), offset 0x170

Register 137: USB Maximum Transmit Data Endpoint 8 (USBTXMAXP8), offset 0x180

Register 138: USB Maximum Transmit Data Endpoint 9 (USBTXMAXP9), offset 0x190

Register 139: USB Maximum Transmit Data Endpoint 10 (USBTXMAXP10), offset 0x1A0

Register 140: USB Maximum Transmit Data Endpoint 11 (USBTXMAXP11), offset 0x1B0

Register 141: USB Maximum Transmit Data Endpoint 12 (USBTXMAXP12), offset 0x1C0

Register 142: USB Maximum Transmit Data Endpoint 13 (USBTXMAXP13), offset 0x1D0

Register 143: USB Maximum Transmit Data Endpoint 14 (USBTXMAXP14), offset 0x1E0

Register 144: USB Maximum Transmit Data Endpoint 15 (USBTXMAXP15), offset 0x1F0

OTG A /
Host

The **USBTXMAXPn** 16-bit register defines the maximum amount of data that can be transferred through the transmit endpoint in a single operation.

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.

OTG B /
Device

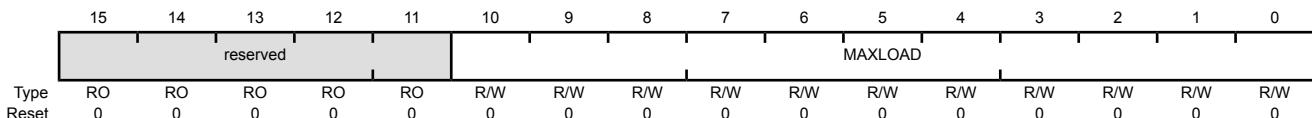
The total amount of data represented by the value written to this register must not exceed the FIFO size for the transmit endpoint, and must not exceed half the FIFO size if double-buffering is required.

If this register is changed after packets have been sent from the endpoint, the transmit endpoint FIFO must be completely flushed (using the **FLUSH** bit in **USBTXCSRLn**) after writing the new value to this register.

Note: **USBTXMAXPn** must be set to an even number of bytes for proper interrupt generation in µDMA Basic Mode.

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	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.
10:0	MAXLOAD	R/W	0x000	Maximum Payload This field specifies the maximum payload in bytes per transaction.

Register 145: USB Control and Status Endpoint 0 Low (USBCSRL0), offset 0x102

OTG A / Host

OTG B / Device

USBCSRL0 is an 8-bit register that provides control and status bits for endpoint 0.

OTG A / 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	NAKTO	STATUS	REQPKT	ERROR	SETUP	STALLED	TXRDY	RXRDY
Reset	R/W 0							

Bit/Field	Name	Type	Reset	Description
7	NAKTO	R/W	0	NAK Timeout Value Description 0 No timeout. 1 Indicates that endpoint 0 is halted following the receipt of NAK responses for longer than the time set by the USBNAKLMT register. Software must clear this bit to allow the endpoint to continue.
6	STATUS	R/W	0	STATUS Packet Value Description 0 No transaction. 1 Initiates a STATUS stage transaction. This bit must be set at the same time as the TXRDY or REQPKT bit is set. Setting this bit ensures that the DT bit is set in the USBCSRH0 register so that a DATA1 packet is used for the STATUS stage transaction. This bit is automatically cleared when the STATUS stage is over.
5	REQPKT	R/W	0	Request Packet Value Description 0 No request. 1 Requests an IN transaction. This bit is cleared when the RXRDY bit is set.

Bit/Field	Name	Type	Reset	Description						
4	ERROR	R/W	0	<p>Error</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No error.</td></tr> <tr> <td>1</td><td>Three attempts have been made to perform a transaction with no response from the peripheral. The EPO bit in the USBTXIS register is also set in this situation.</td></tr> </tbody> </table> <p>Software must clear this bit.</p>	Value	Description	0	No error.	1	Three attempts have been made to perform a transaction with no response from the peripheral. The EPO bit in the USBTXIS register is also set in this situation.
Value	Description									
0	No error.									
1	Three attempts have been made to perform a transaction with no response from the peripheral. The EPO bit in the USBTXIS register is also set in this situation.									
3	SETUP	R/W	0	<p>Setup Packet</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Sends an OUT token.</td></tr> <tr> <td>1</td><td>Sends a SETUP token instead of an OUT token for the transaction. This bit should be set at the same time as the TXRDY bit is set.</td></tr> </tbody> </table> <p>Setting this bit always clears the DT bit in the USBCSRH0 register to send a DATA0 packet.</p>	Value	Description	0	Sends an OUT token.	1	Sends a SETUP token instead of an OUT token for the transaction. This bit should be set at the same time as the TXRDY bit is set.
Value	Description									
0	Sends an OUT token.									
1	Sends a SETUP token instead of an OUT token for the transaction. This bit should be set at the same time as the TXRDY bit is set.									
2	STALLED	R/W	0	<p>Endpoint Stalled</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No handshake has been received.</td></tr> <tr> <td>1</td><td>A STALL handshake has been received.</td></tr> </tbody> </table> <p>Software must clear this bit.</p>	Value	Description	0	No handshake has been received.	1	A STALL handshake has been received.
Value	Description									
0	No handshake has been received.									
1	A STALL handshake has been received.									
1	TXRDY	R/W	0	<p>Transmit Packet Ready</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No transmit packet is ready.</td></tr> <tr> <td>1</td><td>Software sets this bit after loading a data packet into the TX FIFO. The EPO bit in the USBTXIS register is also set in this situation. If both the TXRDY and SETUP bits are set, a setup packet is sent. If just TXRDY is set, an OUT packet is sent.</td></tr> </tbody> </table> <p>This bit is cleared automatically when the data packet has been transmitted.</p>	Value	Description	0	No transmit packet is ready.	1	Software sets this bit after loading a data packet into the TX FIFO. The EPO bit in the USBTXIS register is also set in this situation. If both the TXRDY and SETUP bits are set, a setup packet is sent. If just TXRDY is set, an OUT packet is sent.
Value	Description									
0	No transmit packet is ready.									
1	Software sets this bit after loading a data packet into the TX FIFO. The EPO bit in the USBTXIS register is also set in this situation. If both the TXRDY and SETUP bits are set, a setup packet is sent. If just TXRDY is set, an OUT packet is sent.									
0	RXRDY	R/W	0	<p>Receive Packet Ready</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No received packet has been received.</td></tr> <tr> <td>1</td><td>Indicates that a data packet has been received in the RX FIFO. The EPO bit in the USBTXIS register is also set in this situation.</td></tr> </tbody> </table> <p>Software must clear this bit after the packet has been read from the FIFO to acknowledge that the data has been read from the FIFO.</p>	Value	Description	0	No received packet has been received.	1	Indicates that a data packet has been received in the RX FIFO. The EPO bit in the USBTXIS register is also set in this situation.
Value	Description									
0	No received packet has been received.									
1	Indicates that a data packet has been received in the RX FIFO. The EPO bit in the USBTXIS register is also set in this situation.									

OTG B / Device 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	W1C	W1C	R/W	RO	R/W	R/W	R/W	RO
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	SETENDC	W1C	0	Setup End Clear Writing a 1 to this bit clears the SETEND bit.
6	RXRDYC	W1C	0	RXRDY Clear Writing a 1 to this bit clears the RXRDY bit.
5	STALL	R/W	0	Send Stall Value Description 0 No effect. 1 Terminates the current transaction and transmits the STALL handshake. This bit is cleared automatically after the STALL handshake is transmitted.
4	SETEND	RO	0	Setup End Value Description 0 A control transaction has not ended or ended after the DATAEND bit was set. 1 A control transaction has ended before the DATAEND bit has been set. The EP0 bit in the USBTXIS register is also set in this situation. This bit is cleared by writing a 1 to the SETENDC bit.
3	DATAEND	R/W	0	Data End Value Description 0 No effect. 1 Set this bit in the following situations: <ul style="list-style-type: none">■ When setting TXRDY for the last data packet■ When clearing RXRDY after unloading the last data packet■ When setting TXRDY for a zero-length data packet This bit is cleared automatically.

Bit/Field	Name	Type	Reset	Description
2	STALLED	R/W	0	<p>Endpoint Stalled</p> <p>Value Description</p> <p>0 A STALL handshake has not been transmitted.</p> <p>1 A STALL handshake has been transmitted.</p> <p>Software must clear this bit.</p>
1	TXRDY	R/W	0	<p>Transmit Packet Ready</p> <p>Value Description</p> <p>0 No transmit packet is ready.</p> <p>1 Software sets this bit after loading an IN data packet into the TX FIFO. The EP0 bit in the USBTXIS register is also set in this situation.</p> <p>This bit is cleared automatically when the data packet has been transmitted.</p>
0	RXRDY	RO	0	<p>Receive Packet Ready</p> <p>Value Description</p> <p>0 No data packet has been received.</p> <p>1 A data packet has been received. The EP0 bit in the USBTXIS register is also set in this situation.</p> <p>This bit is cleared by writing a 1 to the RXRDYC bit.</p>

Register 146: USB Control and Status Endpoint 0 High (USBCSRH0), offset 0x103

**OTG A /
Host**

**OTG B /
Device**

USBSR0H is an 8-bit register that provides control and status bits for endpoint 0.

OTG A / 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
Type			reserved			DTWE	DT	FLUSH
Reset	RO 0	RO 0	RO 0	RO 0	RO 0	R/W 0	R/W 0	R/W 0

Bit/Field	Name	Type	Reset	Description
7:3	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	DTWE	R/W	0	Data Toggle Write Enable

Value Description

- 0 The DT bit cannot be written.
- 1 Enables 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 set, this bit may be written with the required setting of the data toggle. If DTWE is Low, this bit cannot be written. Care should be taken when writing to this bit as it should only be changed to RESET USB endpoint 0.
---	----	-----	---	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Bit/Field	Name	Type	Reset	Description
0	FLUSH	R/W	0	Flush FIFO
				Value Description
			0	No effect.
			1	Flushes the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXRDY/RXRDY bit is cleared.
				This bit is automatically cleared after the flush is performed.
Important: This bit should only be set when TXRDY/RXRDY is set. At other times, it may cause data to be corrupted.				

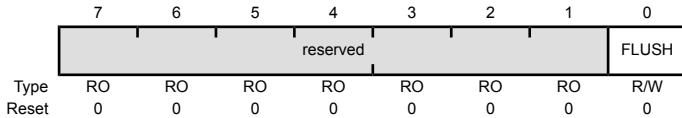
OTG B / Device Mode

USB Control and Status Endpoint 0 High (USBCSRH0)

Base 0x4005.0000

Offset 0x103

Type W1C, reset 0x00



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	R/W	0	Flush FIFO
				Value Description
			0	No effect.
			1	Flushes the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXRDY/RXRDY bit is cleared.
				This bit is automatically cleared after the flush is performed.
Important: This bit should only be set when TXRDY/RXRDY is set. At other times, it may cause data to be corrupted.				

Register 147: USB Receive Byte Count Endpoint 0 (USBCOUNT0), offset 0x108

**OTG A /
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 the RXRDY bit is set.

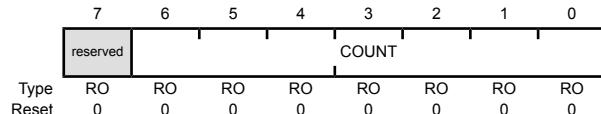
**OTG B /
Device**

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	FIFO Count COUNT is a read-only value that indicates the number of received data bytes in the endpoint 0 FIFO.

Register 148: USB Type Endpoint 0 (USBTYPE0), offset 0x10A

**OTG A /
Host**

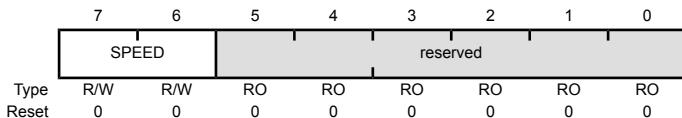
This is an 8-bit register that must be written with the operating speed of the targeted Device being communicated with using endpoint 0.

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	0x0	<p>Operating Speed</p> <p>This field specifies the operating speed of the target Device. If selected, the target is assumed to have the same connection speed as the USB controller.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0 - 0x1</td><td>Reserved</td></tr> <tr> <td>0x2</td><td>Full</td></tr> <tr> <td>0x3</td><td>Low</td></tr> </tbody> </table>	Value	Description	0x0 - 0x1	Reserved	0x2	Full	0x3	Low
Value	Description											
0x0 - 0x1	Reserved											
0x2	Full											
0x3	Low											
5:0	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 149: USB NAK Limit (USBNAKLMT), offset 0x10B

**OTG A /
Host**

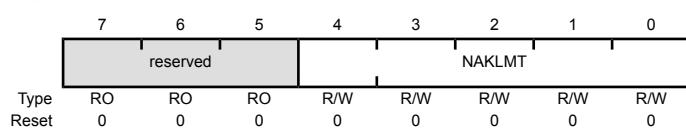
USBNAKLMT is an 8-bit register that sets the number of frames after which endpoint 0 should time out on receiving a stream of NAK responses. (Equivalent settings for other endpoints can be made through their **USBTXINTERVALn** and **USBRXINTERVALn** registers.)

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	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.
4:0	NAKLMT	R/W	0x0	EP0 NAK Limit This field specifies the number of frames after receiving a stream of NAK responses.

**Register 150: USB Transmit Control and Status Endpoint 1 Low
(USBTXCSRL1), offset 0x112**

**Register 151: USB Transmit Control and Status Endpoint 2 Low
(USBTXCSRL2), offset 0x122**

**Register 152: USB Transmit Control and Status Endpoint 3 Low
(USBTXCSRL3), offset 0x132**

**Register 153: USB Transmit Control and Status Endpoint 4 Low
(USBTXCSRL4), offset 0x142**

**Register 154: USB Transmit Control and Status Endpoint 5 Low
(USBTXCSRL5), offset 0x152**

**Register 155: USB Transmit Control and Status Endpoint 6 Low
(USBTXCSRL6), offset 0x162**

**Register 156: USB Transmit Control and Status Endpoint 7 Low
(USBTXCSRL7), offset 0x172**

**Register 157: USB Transmit Control and Status Endpoint 8 Low
(USBTXCSRL8), offset 0x182**

**Register 158: USB Transmit Control and Status Endpoint 9 Low
(USBTXCSRL9), offset 0x192**

**Register 159: USB Transmit Control and Status Endpoint 10 Low
(USBTXCSRL10), offset 0x1A2**

**Register 160: USB Transmit Control and Status Endpoint 11 Low
(USBTXCSRL11), offset 0x1B2**

**Register 161: USB Transmit Control and Status Endpoint 12 Low
(USBTXCSRL12), offset 0x1C2**

**Register 162: USB Transmit Control and Status Endpoint 13 Low
(USBTXCSRL13), offset 0x1D2**

**Register 163: USB Transmit Control and Status Endpoint 14 Low
(USBTXCSRL14), offset 0x1E2**

**Register 164: USB Transmit Control and Status Endpoint 15 Low
(USBTXCSRL15), offset 0x1F2**

OTG A /
Host

USBTXCSRLn is an 8-bit register that provides control and status bits for transfers through the currently selected transmit endpoint.

OTG B /
Device

OTG A / 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
Type	NAKTO	CLRDAT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
Reset	R/W 0							

Bit/Field	Name	Type	Reset	Description
7	NAKTO	R/W	0	NAK Timeout
				Value Description
			0	No timeout.
			1	<i>Bulk endpoints only:</i> Indicates that the transmit endpoint is halted following the receipt of NAK responses for longer than the time set by the NAKLMT field in the USBTXINTERVALn register. Software must clear this bit to allow the endpoint to continue.
6	CLRDAT	R/W	0	Clear Data Toggle Writing a 1 to this bit clears the DT bit in the USBTXCSRn register.
5	STALLED	R/W	0	Endpoint Stalled
				Value Description
			0	A STALL handshake has not been received.
			1	Indicates that a STALL handshake has been received. When this bit is set, any μDMA request that is in progress is stopped, the FIFO is completely flushed, and the TXRDY bit is cleared.
				Software must clear this bit.
4	SETUP	R/W	0	Setup Packet
				Value Description
			0	No SETUP token is sent.
			1	Sends a SETUP token instead of an OUT token for the transaction. This bit should be set at the same time as the TXRDY bit is set.
				Note: Setting this bit also clears the DT bit in the USBTXCSRn register.

Bit/Field	Name	Type	Reset	Description						
3	FLUSH	R/W	0	<p>Flush FIFO</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>Flushes the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset and the TXRDY bit is cleared. The EPn bit in the USBTXIS register is also set in this situation.</td></tr> </tbody> </table> <p>This bit may be set simultaneously with the TXRDY bit to abort the packet that is currently being loaded into the FIFO. Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO.</p> <p>Important: This bit should only be set when the TXRDY bit is set. At other times, it may cause data to be corrupted.</p>	Value	Description	0	No effect.	1	Flushes the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset and the TXRDY bit is cleared. The EPn bit in the USBTXIS register is also set in this situation.
Value	Description									
0	No effect.									
1	Flushes the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset and the TXRDY bit is cleared. The EPn bit in the USBTXIS register is also set in this situation.									
2	ERROR	R/W	0	<p>Error</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No error.</td></tr> <tr> <td>1</td><td>Three attempts have been made to send a packet and no handshake packet has been received. The TXRDY bit is cleared, the EPn bit in the USBTXIS register is set, and the FIFO is completely flushed in this situation.</td></tr> </tbody> </table> <p>Software must clear this bit.</p> <p>Note: This is valid only when the endpoint is operating in Bulk or Interrupt mode.</p>	Value	Description	0	No error.	1	Three attempts have been made to send a packet and no handshake packet has been received. The TXRDY bit is cleared, the EPn bit in the USBTXIS register is set, and the FIFO is completely flushed in this situation.
Value	Description									
0	No error.									
1	Three attempts have been made to send a packet and no handshake packet has been received. The TXRDY bit is cleared, the EPn bit in the USBTXIS register is set, and the FIFO is completely flushed in this situation.									
1	FIFONE	R/W	0	<p>FIFO Not Empty</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The FIFO is empty.</td></tr> <tr> <td>1</td><td>At least one packet is in the transmit FIFO.</td></tr> </tbody> </table>	Value	Description	0	The FIFO is empty.	1	At least one packet is in the transmit FIFO.
Value	Description									
0	The FIFO is empty.									
1	At least one packet is in the transmit FIFO.									
0	TXRDY	R/W	0	<p>Transmit Packet Ready</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No transmit packet is ready.</td></tr> <tr> <td>1</td><td>Software sets this bit after loading a data packet into the TX FIFO.</td></tr> </tbody> </table> <p>This bit is cleared automatically when a data packet has been transmitted. The EPn bit in the USBTXIS register is also set at this point. TXRDY is also automatically cleared prior to loading a second packet into a double-buffered FIFO.</p>	Value	Description	0	No transmit packet is ready.	1	Software sets this bit after loading a data packet into the TX FIFO.
Value	Description									
0	No transmit packet is ready.									
1	Software sets this bit after loading a data packet into the TX FIFO.									

OTG B / 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
Type	reserved	CLRDFT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
Reset	RO 0	R/W 0						

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	CLRDFT	R/W	0	Clear Data Toggle Writing a 1 to this bit clears the DT bit in the USBTXCSRn register.
5	STALLED	R/W	0	Endpoint Stalled Value Description 0 A STALL handshake has not been transmitted. 1 A STALL handshake has been transmitted. The FIFO is flushed and the TXRDY bit is cleared. Software must clear this bit.
4	STALL	R/W	0	Send STALL Value Description 0 No effect. 1 Issues a STALL handshake to an IN token. Software clears this bit to terminate the STALL condition. Note: This bit has no effect in isochronous transfers.
3	FLUSH	R/W	0	Flush FIFO Value Description 0 No effect. 1 Flushes the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset and the TXRDY bit is cleared. The EPn bit in the USBTXIS register is also set in this situation. This bit may be set simultaneously with the TXRDY bit to abort the packet that is currently being loaded into the FIFO. Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO.
Important: This bit should only be set when the TXRDY bit is set. At other times, it may cause data to be corrupted.				

Bit/Field	Name	Type	Reset	Description						
2	UNDRN	R/W	0	<p>Underrun</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No underrun.</td></tr> <tr> <td>1</td><td>An IN token has been received when TXRDY is not set.</td></tr> </tbody> </table> <p>Software must clear this bit.</p>	Value	Description	0	No underrun.	1	An IN token has been received when TXRDY is not set.
Value	Description									
0	No underrun.									
1	An IN token has been received when TXRDY is not set.									
1	FIFONE	R/W	0	<p>FIFO Not Empty</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The FIFO is empty.</td></tr> <tr> <td>1</td><td>At least one packet is in the transmit FIFO.</td></tr> </tbody> </table>	Value	Description	0	The FIFO is empty.	1	At least one packet is in the transmit FIFO.
Value	Description									
0	The FIFO is empty.									
1	At least one packet is in the transmit FIFO.									
0	TXRDY	R/W	0	<p>Transmit Packet Ready</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No transmit packet is ready.</td></tr> <tr> <td>1</td><td>Software sets this bit after loading a data packet into the TX FIFO.</td></tr> </tbody> </table> <p>This bit is cleared automatically when a data packet has been transmitted. The EPn bit in the USBTXIS register is also set at this point. TXRDY is also automatically cleared prior to loading a second packet into a double-buffered FIFO.</p>	Value	Description	0	No transmit packet is ready.	1	Software sets this bit after loading a data packet into the TX FIFO.
Value	Description									
0	No transmit packet is ready.									
1	Software sets this bit after loading a data packet into the TX FIFO.									

**Register 165: USB Transmit Control and Status Endpoint 1 High
(USBTXCSRH1), offset 0x113**

**Register 166: USB Transmit Control and Status Endpoint 2 High
(USBTXCSRH2), offset 0x123**

**Register 167: USB Transmit Control and Status Endpoint 3 High
(USBTXCSRH3), offset 0x133**

**Register 168: USB Transmit Control and Status Endpoint 4 High
(USBTXCSRH4), offset 0x143**

**Register 169: USB Transmit Control and Status Endpoint 5 High
(USBTXCSRH5), offset 0x153**

**Register 170: USB Transmit Control and Status Endpoint 6 High
(USBTXCSRH6), offset 0x163**

**Register 171: USB Transmit Control and Status Endpoint 7 High
(USBTXCSRH7), offset 0x173**

**Register 172: USB Transmit Control and Status Endpoint 8 High
(USBTXCSRH8), offset 0x183**

**Register 173: USB Transmit Control and Status Endpoint 9 High
(USBTXCSRH9), offset 0x193**

**Register 174: USB Transmit Control and Status Endpoint 10 High
(USBTXCSRH10), offset 0x1A3**

**Register 175: USB Transmit Control and Status Endpoint 11 High
(USBTXCSRH11), offset 0x1B3**

**Register 176: USB Transmit Control and Status Endpoint 12 High
(USBTXCSRH12), offset 0x1C3**

**Register 177: USB Transmit Control and Status Endpoint 13 High
(USBTXCSRH13), offset 0x1D3**

**Register 178: USB Transmit Control and Status Endpoint 14 High
(USBTXCSRH14), offset 0x1E3**

**Register 179: USB Transmit Control and Status Endpoint 15 High
(USBTXCSRH15), offset 0x1F3**

OTG A /
Host

USBTXCSRH n is an 8-bit register that provides additional control for transfers through the currently selected transmit endpoint.

OTG B /
Device

OTG A / Host 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
Type	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description						
7	AUTOSET	R/W	0	<p>Auto Set</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The TXRDY bit must be set manually.</td> </tr> <tr> <td>1</td> <td>Enables the TXRDY bit to be automatically set when data of the maximum packet size (value in USBTXMAXPn) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then the TXRDY bit must be set manually.</td> </tr> </tbody> </table>	Value	Description	0	The TXRDY bit must be set manually.	1	Enables the TXRDY bit to be automatically set when data of the maximum packet size (value in USBTXMAXPn) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then the TXRDY bit must be set manually.
Value	Description									
0	The TXRDY bit must be set manually.									
1	Enables the TXRDY bit to be automatically set when data of the maximum packet size (value in USBTXMAXPn) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then the TXRDY bit must be set manually.									
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	<p>Mode</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Enables the endpoint direction as RX.</td> </tr> <tr> <td>1</td> <td>Enables the endpoint direction as TX.</td> </tr> </tbody> </table> <p>Note: This bit only has an effect when the same endpoint FIFO is used for both transmit and receive transactions.</p>	Value	Description	0	Enables the endpoint direction as RX.	1	Enables the endpoint direction as TX.
Value	Description									
0	Enables the endpoint direction as RX.									
1	Enables the endpoint direction as TX.									
4	DMAEN	R/W	0	<p>DMA Request Enable</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Disables the µDMA request for the transmit endpoint.</td> </tr> <tr> <td>1</td> <td>Enables the µDMA request for the transmit endpoint.</td> </tr> </tbody> </table> <p>Note: 3 TX and 3 /RX endpoints can be connected to the µDMA module. If this bit is set for a particular endpoint, the DMAATX, DMABTX, or DMACTX field in the USB DMA Select (USBDMASEL) register must be programmed correspondingly.</p>	Value	Description	0	Disables the µDMA request for the transmit endpoint.	1	Enables the µDMA request for the transmit endpoint.
Value	Description									
0	Disables the µDMA request for the transmit endpoint.									
1	Enables the µDMA request for the transmit endpoint.									
3	FDT	R/W	0	<p>Force Data Toggle</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>Forces the endpoint DT bit to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This bit can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints.</td> </tr> </tbody> </table>	Value	Description	0	No effect.	1	Forces the endpoint DT bit to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This bit can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints.
Value	Description									
0	No effect.									
1	Forces the endpoint DT bit to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This bit can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints.									

Bit/Field	Name	Type	Reset	Description
2	DMAMOD	R/W	0	DMA Request Mode Value Description 0 An interrupt is generated after every µDMA packet transfer. 1 An interrupt is generated only after the entire µDMA transfer is complete. Note: This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.
1	DTWE	R/W	0	Data Toggle Write Enable Value Description 0 The DT bit cannot be written. 1 Enables the current state of the transmit endpoint data to be written (see DT bit). 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. Care should be taken when writing to this bit as it should only be changed to RESET the transmit endpoint.

OTG B / 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	
Type	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO

Bit/Field	Name	Type	Reset	Description
7	AUTOSET	R/W	0	Auto Set Value Description 0 The TXRDY bit must be set manually. 1 Enables the TXRDY bit to be automatically set when data of the maximum packet size (value in USBTXMAXPn) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then the TXRDY bit must be set manually.

Bit/Field	Name	Type	Reset	Description
6	ISO	R/W	0	<p>Isochronous Transfers</p> <p>Value Description</p> <p>0 Enables the transmit endpoint for bulk or interrupt transfers.</p> <p>1 Enables the transmit endpoint for isochronous transfers.</p>
5	MODE	R/W	0	<p>Mode</p> <p>Value Description</p> <p>0 Enables the endpoint direction as RX.</p> <p>1 Enables the endpoint direction as TX.</p> <p>Note: 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>Value Description</p> <p>0 Disables the µDMA request for the transmit endpoint.</p> <p>1 Enables the µDMA request for the transmit endpoint.</p> <p>Note: 3 TX and 3 RX endpoints can be connected to the µDMA module. If this bit is set for a particular endpoint, the DMAATX, DMABTX, or DMACTX field in the USB DMA Select (USBDMASEL) register must be programmed correspondingly.</p>
3	FDT	R/W	0	<p>Force Data Toggle</p> <p>Value Description</p> <p>0 No effect.</p> <p>1 Forces the endpoint DT bit to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This bit 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>Value Description</p> <p>0 An interrupt is generated after every µDMA packet transfer.</p> <p>1 An interrupt is generated only after the entire µDMA transfer is complete.</p> <p>Note: This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.</p>
1:0	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 180: USB Maximum Receive Data Endpoint 1 (USBRXMAXP1), offset 0x114

Register 181: USB Maximum Receive Data Endpoint 2 (USBRXMAXP2), offset 0x124

Register 182: USB Maximum Receive Data Endpoint 3 (USBRXMAXP3), offset 0x134

Register 183: USB Maximum Receive Data Endpoint 4 (USBRXMAXP4), offset 0x144

Register 184: USB Maximum Receive Data Endpoint 5 (USBRXMAXP5), offset 0x154

Register 185: USB Maximum Receive Data Endpoint 6 (USBRXMAXP6), offset 0x164

Register 186: USB Maximum Receive Data Endpoint 7 (USBRXMAXP7), offset 0x174

Register 187: USB Maximum Receive Data Endpoint 8 (USBRXMAXP8), offset 0x184

Register 188: USB Maximum Receive Data Endpoint 9 (USBRXMAXP9), offset 0x194

Register 189: USB Maximum Receive Data Endpoint 10 (USBRXMAXP10), offset 0x1A4

Register 190: USB Maximum Receive Data Endpoint 11 (USBRXMAXP11), offset 0x1B4

Register 191: USB Maximum Receive Data Endpoint 12 (USBRXMAXP12), offset 0x1C4

Register 192: USB Maximum Receive Data Endpoint 13 (USBRXMAXP13), offset 0x1D4

Register 193: USB Maximum Receive Data Endpoint 14 (USBRXMAXP14), offset 0x1E4

Register 194: USB Maximum Receive Data Endpoint 15 (USBRXMAXP15), offset 0x1F4

OTG A /
Host

The **USBRXMAXPn** is a 16-bit register which defines the maximum amount of data that can be transferred through the selected receive endpoint in a single operation.

OTG B /
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 must not exceed the FIFO size for the receive endpoint, and must not exceed half the FIFO size if double-buffering is required.

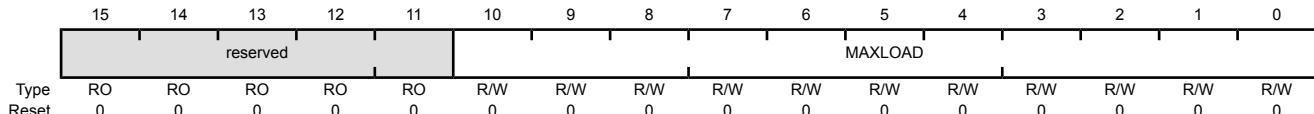
Note: **USBRXMAXPn** must be set to an even number of bytes for proper interrupt generation in µDMA Basic mode.

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	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.
10:0	MAXLOAD	R/W	0x000	Maximum Payload The maximum payload in bytes per transaction.

Register 195: USB Receive Control and Status Endpoint 1 Low (USBRXCSR1), offset 0x116

Register 196: USB Receive Control and Status Endpoint 2 Low (USBRXCSR2), offset 0x126

Register 197: USB Receive Control and Status Endpoint 3 Low (USBRXCSR3), offset 0x136

Register 198: USB Receive Control and Status Endpoint 4 Low (USBRXCSR4), offset 0x146

Register 199: USB Receive Control and Status Endpoint 5 Low (USBRXCSR5), offset 0x156

Register 200: USB Receive Control and Status Endpoint 6 Low (USBRXCSR6), offset 0x166

Register 201: USB Receive Control and Status Endpoint 7 Low (USBRXCSR7), offset 0x176

Register 202: USB Receive Control and Status Endpoint 8 Low (USBRXCSR8), offset 0x186

Register 203: USB Receive Control and Status Endpoint 9 Low (USBRXCSR9), offset 0x196

Register 204: USB Receive Control and Status Endpoint 10 Low (USBRXCSR10), offset 0x1A6

Register 205: USB Receive Control and Status Endpoint 11 Low (USBRXCSR11), offset 0x1B6

Register 206: USB Receive Control and Status Endpoint 12 Low (USBRXCSR12), offset 0x1C6

Register 207: USB Receive Control and Status Endpoint 13 Low (USBRXCSR13), offset 0x1D6

Register 208: USB Receive Control and Status Endpoint 14 Low (USBRXCSR14), offset 0x1E6

Register 209: USB Receive Control and Status Endpoint 15 Low (USBRXCSR15), offset 0x1F6

OTG A /
Host

USBRXCSR_n is an 8-bit register that provides control and status bits for transfers through the currently selected receive endpoint.

OTG B /
Device

OTG A / Host Mode**USB Receive Control and Status Endpoint 1 Low (USBRXCSRL1)**

Base 0x4005.0000
Offset 0x116
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
Type	CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
Reset	W1C	R/W	R/W	R/W	R/W	R/W	RO	R/W

Bit/Field	Name	Type	Reset	Description
7	CLRDT	W1C	0	Clear Data Toggle Writing a 1 to this bit clears the DT bit in the USBRXCSRn register.
6	STALLED	R/W	0	Endpoint Stalled Value Description 0 A STALL handshake has not been received. 1 A STALL handshake has been received. The EPn bit in the USBRXIS register is also set. Software must clear this bit.
5	REQPKT	R/W	0	Request Packet Value Description 0 No request. 1 Requests an IN transaction. This bit is cleared when RXRDY is set.
4	FLUSH	R/W	0	Flush FIFO Value Description 0 No effect. 1 Flushes the next packet to be read from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared. Note that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO.

Important: This bit should only be set when the RXRDY bit is set. At other times, it may cause data to be corrupted.

Bit/Field	Name	Type	Reset	Description						
3	DATAERR / NAKTO	R/W	0	<p>Data Error / NAK Timeout</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Normal operation.</td></tr> <tr> <td>1</td><td> <i>Isochronous endpoints only:</i> Indicates that RXRDY is set and the data packet has a CRC or bit-stuff error. This bit is cleared when RXRDY is cleared. <i>Bulk endpoints only:</i> Indicates that the receive endpoint is halted following the receipt of NAK responses for longer than the time set by the NAKLMT field in the USBRXINTERVALn register. Software must clear this bit to allow the endpoint to continue. </td></tr> </tbody> </table>	Value	Description	0	Normal operation.	1	<i>Isochronous endpoints only:</i> Indicates that RXRDY is set and the data packet has a CRC or bit-stuff error. This bit is cleared when RXRDY is cleared. <i>Bulk endpoints only:</i> Indicates that the receive endpoint is halted following the receipt of NAK responses for longer than the time set by the NAKLMT field in the USBRXINTERVALn register. Software must clear this bit to allow the endpoint to continue.
Value	Description									
0	Normal operation.									
1	<i>Isochronous endpoints only:</i> Indicates that RXRDY is set and the data packet has a CRC or bit-stuff error. This bit is cleared when RXRDY is cleared. <i>Bulk endpoints only:</i> Indicates that the receive endpoint is halted following the receipt of NAK responses for longer than the time set by the NAKLMT field in the USBRXINTERVALn register. Software must clear this bit to allow the endpoint to continue.									
2	ERROR	R/W	0	<p>Error</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No error.</td></tr> <tr> <td>1</td><td>Three attempts have been made to receive a packet and no data packet has been received. The EPn bit in the USBRXIS register is set in this situation.</td></tr> </tbody> </table> <p>Software must clear this bit.</p> <p>Note: This bit is only valid when the receive endpoint is operating in Bulk or Interrupt mode. In Isochronous mode, it always returns zero.</p>	Value	Description	0	No error.	1	Three attempts have been made to receive a packet and no data packet has been received. The EPn bit in the USBRXIS register is set in this situation.
Value	Description									
0	No error.									
1	Three attempts have been made to receive a packet and no data packet has been received. The EPn bit in the USBRXIS register is set in this situation.									
1	FULL	RO	0	<p>FIFO Full</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The receive FIFO is not full.</td></tr> <tr> <td>1</td><td>No more packets can be loaded into the receive FIFO.</td></tr> </tbody> </table>	Value	Description	0	The receive FIFO is not full.	1	No more packets can be loaded into the receive FIFO.
Value	Description									
0	The receive FIFO is not full.									
1	No more packets can be loaded into the receive FIFO.									
0	RXRDY	R/W	0	<p>Receive Packet Ready</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No data packet has been received.</td></tr> <tr> <td>1</td><td>A data packet has been received. The EPn bit in the USBRXIS register is also set in this situation.</td></tr> </tbody> </table> <p>If the AUTOCLR bit in the USBRXCSRn register is set, then this bit is automatically cleared when a packet of USBRXMAXPn bytes has been unloaded from the receive FIFO. If the AUTOCLR bit is clear, or if packets of less than the maximum packet size are unloaded, then software must clear this bit manually when the packet has been unloaded from the receive FIFO.</p>	Value	Description	0	No data packet has been received.	1	A data packet has been received. The EPn bit in the USBRXIS register is also set in this situation.
Value	Description									
0	No data packet has been received.									
1	A data packet has been received. The EPn bit in the USBRXIS register is also set in this situation.									

OTG B / Device Mode

USB Receive Control and Status Endpoint 1 Low (USBRXCSRL1)

Base 0x4005.0000
Offset 0x116
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
Type	CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
Reset	W1C	R/W	R/W	R/W	RO	R/W	RO	R/W

Bit/Field	Name	Type	Reset	Description
7	CLRDT	W1C	0	Clear Data Toggle Writing a 1 to this bit clears the DT bit in the USBRXCSRn register.
6	STALLED	R/W	0	Endpoint Stalled Value Description 0 A STALL handshake has not been transmitted. 1 A STALL handshake has been transmitted. Software must clear this bit.
5	STALL	R/W	0	Send STALL Value Description 0 No effect. 1 Issues a STALL handshake. Software must clear this bit to terminate the STALL condition. Note: This bit has no effect where the endpoint is being used for isochronous transfers.
4	FLUSH	R/W	0	Flush FIFO Value Description 0 No effect. 1 Flushes the next packet from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared. 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 that if the FIFO is double-buffered, FLUSH may have to be set twice to completely clear the FIFO.
Important: This bit should only be set when the RXRDY bit is set. At other times, it may cause data to be corrupted.				

Bit/Field	Name	Type	Reset	Description						
3	DATAERR	RO	0	<p>Data Error</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Normal operation.</td></tr> <tr> <td>1</td><td>Indicates that RXRDY is set and the data packet has a CRC or bit-stuff error.</td></tr> </tbody> </table> <p>This bit is cleared when RXRDY is cleared.</p> <p>Note: This bit is only valid when the endpoint is operating in Isochronous mode. In Bulk mode, it always returns zero.</p>	Value	Description	0	Normal operation.	1	Indicates that RXRDY is set and the data packet has a CRC or bit-stuff error.
Value	Description									
0	Normal operation.									
1	Indicates that RXRDY is set and the data packet has a CRC or bit-stuff error.									
2	OVER	R/W	0	<p>Overrun</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No overrun error.</td></tr> <tr> <td>1</td><td>Indicates that an OUT packet cannot be loaded into the receive FIFO.</td></tr> </tbody> </table> <p>Software must clear this bit.</p> <p>Note: This bit is only valid when the endpoint is operating in Isochronous mode. In Bulk mode, it always returns zero.</p>	Value	Description	0	No overrun error.	1	Indicates that an OUT packet cannot be loaded into the receive FIFO.
Value	Description									
0	No overrun error.									
1	Indicates that an OUT packet cannot be loaded into the receive FIFO.									
1	FULL	RO	0	<p>FIFO Full</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The receive FIFO is not full.</td></tr> <tr> <td>1</td><td>No more packets can be loaded into the receive FIFO.</td></tr> </tbody> </table>	Value	Description	0	The receive FIFO is not full.	1	No more packets can be loaded into the receive FIFO.
Value	Description									
0	The receive FIFO is not full.									
1	No more packets can be loaded into the receive FIFO.									
0	RXRDY	R/W	0	<p>Receive Packet Ready</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>No data packet has been received.</td></tr> <tr> <td>1</td><td>A data packet has been received. The EPn bit in the USBRXIS register is also set in this situation.</td></tr> </tbody> </table> <p>If the AUTOCLR bit in the USBRXCSRHn register is set, then this bit is automatically cleared when a packet of USBRXMAXPn bytes has been unloaded from the receive FIFO. If the AUTOCLR bit is clear, or if packets of less than the maximum packet size are unloaded, then software must clear this bit manually when the packet has been unloaded from the receive FIFO.</p>	Value	Description	0	No data packet has been received.	1	A data packet has been received. The EPn bit in the USBRXIS register is also set in this situation.
Value	Description									
0	No data packet has been received.									
1	A data packet has been received. The EPn bit in the USBRXIS register is also set in this situation.									

**Register 210: USB Receive Control and Status Endpoint 1 High
(USBRXCSRH1), offset 0x117**

**Register 211: USB Receive Control and Status Endpoint 2 High
(USBRXCSRH2), offset 0x127**

**Register 212: USB Receive Control and Status Endpoint 3 High
(USBRXCSRH3), offset 0x137**

**Register 213: USB Receive Control and Status Endpoint 4 High
(USBRXCSRH4), offset 0x147**

**Register 214: USB Receive Control and Status Endpoint 5 High
(USBRXCSRH5), offset 0x157**

**Register 215: USB Receive Control and Status Endpoint 6 High
(USBRXCSRH6), offset 0x167**

**Register 216: USB Receive Control and Status Endpoint 7 High
(USBRXCSRH7), offset 0x177**

**Register 217: USB Receive Control and Status Endpoint 8 High
(USBRXCSRH8), offset 0x187**

**Register 218: USB Receive Control and Status Endpoint 9 High
(USBRXCSRH9), offset 0x197**

**Register 219: USB Receive Control and Status Endpoint 10 High
(USBRXCSRH10), offset 0x1A7**

**Register 220: USB Receive Control and Status Endpoint 11 High
(USBRXCSRH11), offset 0x1B7**

**Register 221: USB Receive Control and Status Endpoint 12 High
(USBRXCSRH12), offset 0x1C7**

**Register 222: USB Receive Control and Status Endpoint 13 High
(USBRXCSRH13), offset 0x1D7**

**Register 223: USB Receive Control and Status Endpoint 14 High
(USBRXCSRH14), offset 0x1E7**

**Register 224: USB Receive Control and Status Endpoint 15 High
(USBRXCSRH15), offset 0x1F7**

OTG A /
Host

USBRXCSRH n is an 8-bit register that provides additional control and status bits for transfers through the currently selected receive endpoint.

OTG B /
Device

OTG A / Host Mode**USB Receive Control and Status Endpoint 1 High (USBRXCSRH1)**

Base 0x4005.0000

Offset 0x117

Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
Type	R/W	R/W	R/W	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	AUTOCL	R/W	0	Auto Clear Value Description 0 No effect. 1 Enables the RXRDY bit to be automatically cleared when a packet of USBRXMAXPn 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 μDMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of the value of the MAXLOAD field in the USBRXMAXPn register, see “DMA Operation” on page 928.
6	AUTORQ	R/W	0	Auto Request Value Description 0 No effect. 1 Enables the REQPKT bit to be 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 Value Description 0 Disables the μDMA request for the receive endpoint. 1 Enables the μDMA request for the receive endpoint. Note: 3 TX and 3 RX endpoints can be connected to the μDMA module. If this bit is set for a particular endpoint, the DMAARX, DMABRX, or DMACRX field in the USB DMA Select (USBDMASEL) register must be programmed correspondingly.
4	PIDERR	RO	0	PID Error Value Description 0 No error. 1 Indicates a PID error in the received packet of an isochronous transaction. This bit is ignored in bulk or interrupt transactions.

Bit/Field	Name	Type	Reset	Description
3	DMAMOD	R/W	0	DMA Request Mode Value Description 0 An interrupt is generated after every µDMA packet transfer. 1 An interrupt is generated only after the entire µDMA transfer is complete. Note: This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.
2	DTWE	RO	0	Data Toggle Write Enable Value Description 0 The DT bit cannot be written. 1 Enables the current state of the receive endpoint data to be written (see DT bit). 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 receive 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. Care should be taken when writing to this bit as it should only be changed to RESET the receive endpoint.
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.

OTG B / Device Mode

USB Receive Control and Status Endpoint 1 High (USBRXCSRH1)

Base 0x4005.0000
Offset 0x117

Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
Type	R/W	R/W	R/W	R/W	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0
AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD	reserved			

Bit/Field	Name	Type	Reset	Description						
7	AUTOCL	R/W	0	<p>Auto Clear</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>Enables the RXRDY bit to be automatically cleared when a packet of USBRXMAXPn 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 μDMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of the value of the MAXLOAD field in the USBRXMAXPn register, see “DMA Operation” on page 928.</td></tr> </tbody> </table>	Value	Description	0	No effect.	1	Enables the RXRDY bit to be automatically cleared when a packet of USBRXMAXPn 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 μDMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of the value of the MAXLOAD field in the USBRXMAXPn register, see “DMA Operation” on page 928.
Value	Description									
0	No effect.									
1	Enables the RXRDY bit to be automatically cleared when a packet of USBRXMAXPn 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 μDMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of the value of the MAXLOAD field in the USBRXMAXPn register, see “DMA Operation” on page 928.									
6	ISO	R/W	0	<p>Isochronous Transfers</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Enables the receive endpoint for isochronous transfers.</td></tr> <tr> <td>1</td><td>Enables the receive endpoint for bulk/interrupt transfers.</td></tr> </tbody> </table>	Value	Description	0	Enables the receive endpoint for isochronous transfers.	1	Enables the receive endpoint for bulk/interrupt transfers.
Value	Description									
0	Enables the receive endpoint for isochronous transfers.									
1	Enables the receive endpoint for bulk/interrupt transfers.									
5	DMAEN	R/W	0	<p>DMA Request Enable</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Disables the μDMA request for the receive endpoint.</td></tr> <tr> <td>1</td><td>Enables the μDMA request for the receive endpoint.</td></tr> </tbody> </table> <p>Note: 3 TX and 3 RX endpoints can be connected to the μDMA module. If this bit is set for a particular endpoint, the DMAARX, DMABRX, or DMACRX field in the USB DMA Select (USBDMASEL) register must be programmed correspondingly.</p>	Value	Description	0	Disables the μDMA request for the receive endpoint.	1	Enables the μDMA request for the receive endpoint.
Value	Description									
0	Disables the μDMA request for the receive endpoint.									
1	Enables the μDMA request for the receive endpoint.									
4	DISNYET / PIDERR	R/W	0	<p>Disable NYET / PID Error</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> <p><i>For bulk or interrupt transactions:</i> Disables the sending of NYET handshakes. When this bit is set, all successfully received packets are acknowledged, including at the point at which the FIFO becomes full.</p> <p><i>For isochronous transactions:</i> Indicates a PID error in the received packet.</p> </td></tr> </tbody> </table>	Value	Description	0	No effect.	1	<p><i>For bulk or interrupt transactions:</i> Disables the sending of NYET handshakes. When this bit is set, all successfully received packets are acknowledged, including at the point at which the FIFO becomes full.</p> <p><i>For isochronous transactions:</i> Indicates a PID error in the received packet.</p>
Value	Description									
0	No effect.									
1	<p><i>For bulk or interrupt transactions:</i> Disables the sending of NYET handshakes. When this bit is set, all successfully received packets are acknowledged, including at the point at which the FIFO becomes full.</p> <p><i>For isochronous transactions:</i> Indicates a PID error in the received packet.</p>									
3	DMAMOD	R/W	0	<p>DMA Request Mode</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>An interrupt is generated after every μDMA packet transfer.</td></tr> <tr> <td>1</td><td>An interrupt is generated only after the entire μDMA transfer is complete.</td></tr> </tbody> </table> <p>Note: This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.</p>	Value	Description	0	An interrupt is generated after every μDMA packet transfer.	1	An interrupt is generated only after the entire μDMA transfer is complete.
Value	Description									
0	An interrupt is generated after every μDMA packet transfer.									
1	An interrupt is generated only after the entire μDMA transfer is complete.									

Bit/Field	Name	Type	Reset	Description
2:0	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 225: USB Receive Byte Count Endpoint 1 (USBRXCOUNT1), offset 0x118**
- Register 226: USB Receive Byte Count Endpoint 2 (USBRXCOUNT2), offset 0x128**
- Register 227: USB Receive Byte Count Endpoint 3 (USBRXCOUNT3), offset 0x138**
- Register 228: USB Receive Byte Count Endpoint 4 (USBRXCOUNT4), offset 0x148**
- Register 229: USB Receive Byte Count Endpoint 5 (USBRXCOUNT5), offset 0x158**
- Register 230: USB Receive Byte Count Endpoint 6 (USBRXCOUNT6), offset 0x168**
- Register 231: USB Receive Byte Count Endpoint 7 (USBRXCOUNT7), offset 0x178**
- Register 232: USB Receive Byte Count Endpoint 8 (USBRXCOUNT8), offset 0x188**
- Register 233: USB Receive Byte Count Endpoint 9 (USBRXCOUNT9), offset 0x198**
- Register 234: USB Receive Byte Count Endpoint 10 (USBRXCOUNT10), offset 0x1A8**
- Register 235: USB Receive Byte Count Endpoint 11 (USBRXCOUNT11), offset 0x1B8**
- Register 236: USB Receive Byte Count Endpoint 12 (USBRXCOUNT12), offset 0x1C8**
- Register 237: USB Receive Byte Count Endpoint 13 (USBRXCOUNT13), offset 0x1D8**
- Register 238: USB Receive Byte Count Endpoint 14 (USBRXCOUNT14), offset 0x1E8**
- Register 239: USB Receive Byte Count Endpoint 15 (USBRXCOUNT15), offset 0x1F8**

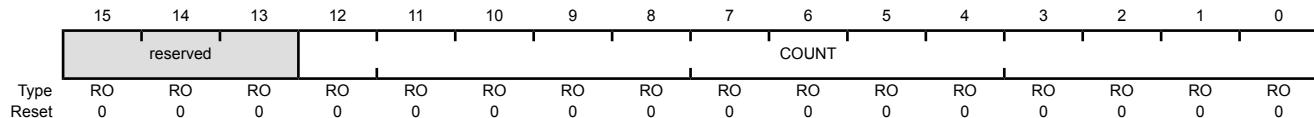
OTG A /
Host
OTG B /
Device

Note: The value returned changes as the FIFO is unloaded and is only valid while the RXRDY bit in the **USBRXCSRLn** register is set.

USBRXCOUNTn 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	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.
12:0	COUNT	RO	0x000	Receive Packet Count Indicates the number of bytes in the receive packet.

Register 240: USB Host Transmit Configure Type Endpoint 1 (USBTXTYPE1), offset 0x11A

Register 241: USB Host Transmit Configure Type Endpoint 2 (USBTXTYPE2), offset 0x12A

Register 242: USB Host Transmit Configure Type Endpoint 3 (USBTXTYPE3), offset 0x13A

Register 243: USB Host Transmit Configure Type Endpoint 4 (USBTXTYPE4), offset 0x14A

Register 244: USB Host Transmit Configure Type Endpoint 5 (USBTXTYPE5), offset 0x15A

Register 245: USB Host Transmit Configure Type Endpoint 6 (USBTXTYPE6), offset 0x16A

Register 246: USB Host Transmit Configure Type Endpoint 7 (USBTXTYPE7), offset 0x17A

Register 247: USB Host Transmit Configure Type Endpoint 8 (USBTXTYPE8), offset 0x18A

Register 248: USB Host Transmit Configure Type Endpoint 9 (USBTXTYPE9), offset 0x19A

Register 249: USB Host Transmit Configure Type Endpoint 10 (USBTXTYPE10), offset 0x1AA

Register 250: USB Host Transmit Configure Type Endpoint 11 (USBTXTYPE11), offset 0x1BA

Register 251: USB Host Transmit Configure Type Endpoint 12 (USBTXTYPE12), offset 0x1CA

Register 252: USB Host Transmit Configure Type Endpoint 13 (USBTXTYPE13), offset 0x1DA

Register 253: USB Host Transmit Configure Type Endpoint 14 (USBTXTYPE14), offset 0x1EA

Register 254: USB Host Transmit Configure Type Endpoint 15 (USBTXTYPE15), offset 0x1FA

OTG A /
Host

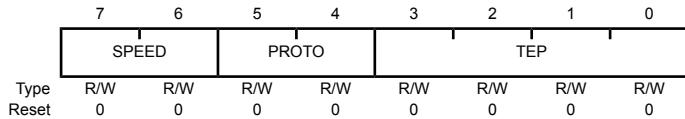
USBTXTYPEn is an 8-bit register that must be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected transmit endpoint, and its operating speed.

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	0x0	<p>Operating Speed This bit field specifies the operating speed of the target Device:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Default The target is assumed to be using the same connection speed as the USB controller.</td></tr> <tr> <td>0x1</td><td>Reserved</td></tr> <tr> <td>0x2</td><td>Full</td></tr> <tr> <td>0x3</td><td>Low</td></tr> </tbody> </table>	Value	Description	0x0	Default The target is assumed to be using the same connection speed as the USB controller.	0x1	Reserved	0x2	Full	0x3	Low
Value	Description													
0x0	Default The target is assumed to be using the same connection speed as the USB controller.													
0x1	Reserved													
0x2	Full													
0x3	Low													
5:4	PROTO	R/W	0x0	<p>Protocol Software must configure this bit field to select the required protocol for the transmit endpoint:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Control</td></tr> <tr> <td>0x1</td><td>Isochronous</td></tr> <tr> <td>0x2</td><td>Bulk</td></tr> <tr> <td>0x3</td><td>Interrupt</td></tr> </tbody> </table>	Value	Description	0x0	Control	0x1	Isochronous	0x2	Bulk	0x3	Interrupt
Value	Description													
0x0	Control													
0x1	Isochronous													
0x2	Bulk													
0x3	Interrupt													
3:0	TEP	R/W	0x0	<p>Target Endpoint Number Software must configure this value to the endpoint number contained in the transmit endpoint descriptor returned to the USB controller during Device enumeration.</p>										

Register 255: USB Host Transmit Interval Endpoint 1 (USBTXINTERVAL1), offset 0x11B

Register 256: USB Host Transmit Interval Endpoint 2 (USBTXINTERVAL2), offset 0x12B

Register 257: USB Host Transmit Interval Endpoint 3 (USBTXINTERVAL3), offset 0x13B

Register 258: USB Host Transmit Interval Endpoint 4 (USBTXINTERVAL4), offset 0x14B

Register 259: USB Host Transmit Interval Endpoint 5 (USBTXINTERVAL5), offset 0x15B

Register 260: USB Host Transmit Interval Endpoint 6 (USBTXINTERVAL6), offset 0x16B

Register 261: USB Host Transmit Interval Endpoint 7 (USBTXINTERVAL7), offset 0x17B

Register 262: USB Host Transmit Interval Endpoint 8 (USBTXINTERVAL8), offset 0x18B

Register 263: USB Host Transmit Interval Endpoint 9 (USBTXINTERVAL9), offset 0x19B

Register 264: USB Host Transmit Interval Endpoint 10 (USBTXINTERVAL10), offset 0x1AB

Register 265: USB Host Transmit Interval Endpoint 11 (USBTXINTERVAL11), offset 0x1BB

Register 266: USB Host Transmit Interval Endpoint 12 (USBTXINTERVAL12), offset 0x1CB

Register 267: USB Host Transmit Interval Endpoint 13 (USBTXINTERVAL13), offset 0x1DB

Register 268: USB Host Transmit Interval Endpoint 14 (USBTXINTERVAL14), offset 0x1EB

Register 269: USB Host Transmit Interval Endpoint 15 (USBTXINTERVAL15), offset 0x1FB

OTG A /
Host

USBTXINTERVALn 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 defines the number of frames after which the endpoint should time out on receiving a stream of NAK responses.

The USBTXINTERVALn register value defines a number of frames, as follows:

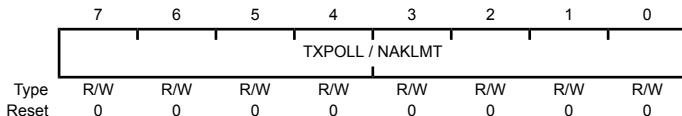
Transfer Type	Speed	Valid values (m)	Interpretation
Interrupt	Low-Speed or Full-Speed	0x01 – 0xFF	The polling interval is m frames.
Isochronous	Full-Speed	0x01 – 0x10	The polling interval is $2^{(m-1)}$ frames.
Bulk	Full-Speed	0x02 – 0x10	The NAK Limit is $2^{(m-1)}$ frames. A value of 0 or 1 disables 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 The polling interval for interrupt/isochronous transfers; the NAK limit for bulk transfers. See table above for valid entries; other values are reserved.

Register 270: USB Host Configure Receive Type Endpoint 1 (USBRXTYPE1), offset 0x11C

Register 271: USB Host Configure Receive Type Endpoint 2 (USBRXTYPE2), offset 0x12C

Register 272: USB Host Configure Receive Type Endpoint 3 (USBRXTYPE3), offset 0x13C

Register 273: USB Host Configure Receive Type Endpoint 4 (USBRXTYPE4), offset 0x14C

Register 274: USB Host Configure Receive Type Endpoint 5 (USBRXTYPE5), offset 0x15C

Register 275: USB Host Configure Receive Type Endpoint 6 (USBRXTYPE6), offset 0x16C

Register 276: USB Host Configure Receive Type Endpoint 7 (USBRXTYPE7), offset 0x17C

Register 277: USB Host Configure Receive Type Endpoint 8 (USBRXTYPE8), offset 0x18C

Register 278: USB Host Configure Receive Type Endpoint 9 (USBRXTYPE9), offset 0x19C

Register 279: USB Host Configure Receive Type Endpoint 10 (USBRXTYPE10), offset 0x1AC

Register 280: USB Host Configure Receive Type Endpoint 11 (USBRXTYPE11), offset 0x1BC

Register 281: USB Host Configure Receive Type Endpoint 12 (USBRXTYPE12), offset 0x1CC

Register 282: USB Host Configure Receive Type Endpoint 13 (USBRXTYPE13), offset 0x1DC

Register 283: USB Host Configure Receive Type Endpoint 14 (USBRXTYPE14), offset 0x1EC

Register 284: USB Host Configure Receive Type Endpoint 15 (USBRXTYPE15), offset 0x1FC

OTG A /
Host

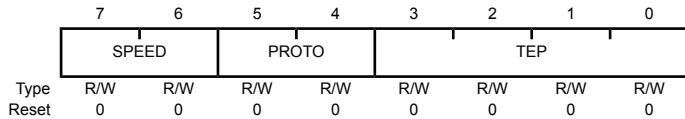
USBRXTYPEn is an 8-bit register that must be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected receive endpoint, and its operating speed.

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	0x0	<p>Operating Speed This bit field specifies the operating speed of the target Device:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Default The target is assumed to be using the same connection speed as the USB controller.</td></tr> <tr> <td>0x1</td><td>Reserved</td></tr> <tr> <td>0x2</td><td>Full</td></tr> <tr> <td>0x3</td><td>Low</td></tr> </tbody> </table>	Value	Description	0x0	Default The target is assumed to be using the same connection speed as the USB controller.	0x1	Reserved	0x2	Full	0x3	Low
Value	Description													
0x0	Default The target is assumed to be using the same connection speed as the USB controller.													
0x1	Reserved													
0x2	Full													
0x3	Low													
5:4	PROTO	R/W	0x0	<p>Protocol Software must configure this bit field to select the required protocol for the receive endpoint:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Control</td></tr> <tr> <td>0x1</td><td>Isochronous</td></tr> <tr> <td>0x2</td><td>Bulk</td></tr> <tr> <td>0x3</td><td>Interrupt</td></tr> </tbody> </table>	Value	Description	0x0	Control	0x1	Isochronous	0x2	Bulk	0x3	Interrupt
Value	Description													
0x0	Control													
0x1	Isochronous													
0x2	Bulk													
0x3	Interrupt													
3:0	TEP	R/W	0x0	<p>Target Endpoint Number Software must set this value to the endpoint number contained in the receive endpoint descriptor returned to the USB controller during Device enumeration.</p>										

**Register 285: USB Host Receive Polling Interval Endpoint 1
(USBRXINTERVAL1), offset 0x11D**

**Register 286: USB Host Receive Polling Interval Endpoint 2
(USBRXINTERVAL2), offset 0x12D**

**Register 287: USB Host Receive Polling Interval Endpoint 3
(USBRXINTERVAL3), offset 0x13D**

**Register 288: USB Host Receive Polling Interval Endpoint 4
(USBRXINTERVAL4), offset 0x14D**

**Register 289: USB Host Receive Polling Interval Endpoint 5
(USBRXINTERVAL5), offset 0x15D**

**Register 290: USB Host Receive Polling Interval Endpoint 6
(USBRXINTERVAL6), offset 0x16D**

**Register 291: USB Host Receive Polling Interval Endpoint 7
(USBRXINTERVAL7), offset 0x17D**

**Register 292: USB Host Receive Polling Interval Endpoint 8
(USBRXINTERVAL8), offset 0x18D**

**Register 293: USB Host Receive Polling Interval Endpoint 9
(USBRXINTERVAL9), offset 0x19D**

**Register 294: USB Host Receive Polling Interval Endpoint 10
(USBRXINTERVAL10), offset 0x1AD**

**Register 295: USB Host Receive Polling Interval Endpoint 11
(USBRXINTERVAL11), offset 0x1BD**

**Register 296: USB Host Receive Polling Interval Endpoint 12
(USBRXINTERVAL12), offset 0x1CD**

**Register 297: USB Host Receive Polling Interval Endpoint 13
(USBRXINTERVAL13), offset 0x1DD**

**Register 298: USB Host Receive Polling Interval Endpoint 14
(USBRXINTERVAL14), offset 0x1ED**

**Register 299: USB Host Receive Polling Interval Endpoint 15
(USBRXINTERVAL15), offset 0x1FD**

OTG A /
Host

USBRXINTERVALn 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 defines the number of frames after which the endpoint should time out on receiving a stream of NAK responses.

The USBTXINTERVALn register value defines a number of frames, as follows:

Transfer Type	Speed	Valid values (m)	Interpretation
Interrupt	Low-Speed or Full-Speed	0x01 – 0xFF	The polling interval is m frames.
Isochronous	Full-Speed	0x01 – 0x10	The polling interval is $2^{(m-1)}$ frames.

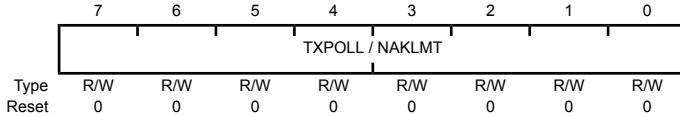
Transfer Type	Speed	Valid values (m)	Interpretation
Bulk	Full-Speed	0x02 – 0x10	The NAK Limit is $2^{(m-1)}$ frames. A value of 0 or 1 disables the NAK timeout function.

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 The polling interval for interrupt/isochronous transfers; the NAK limit for bulk transfers. See table above for valid entries; other values are reserved.

**Register 300: USB Request Packet Count in Block Transfer Endpoint 1
(USBRQPKTCOUNT1), offset 0x304**

**Register 301: USB Request Packet Count in Block Transfer Endpoint 2
(USBRQPKTCOUNT2), offset 0x308**

**Register 302: USB Request Packet Count in Block Transfer Endpoint 3
(USBRQPKTCOUNT3), offset 0x30C**

**Register 303: USB Request Packet Count in Block Transfer Endpoint 4
(USBRQPKTCOUNT4), offset 0x310**

**Register 304: USB Request Packet Count in Block Transfer Endpoint 5
(USBRQPKTCOUNT5), offset 0x314**

**Register 305: USB Request Packet Count in Block Transfer Endpoint 6
(USBRQPKTCOUNT6), offset 0x318**

**Register 306: USB Request Packet Count in Block Transfer Endpoint 7
(USBRQPKTCOUNT7), offset 0x31C**

**Register 307: USB Request Packet Count in Block Transfer Endpoint 8
(USBRQPKTCOUNT8), offset 0x320**

**Register 308: USB Request Packet Count in Block Transfer Endpoint 9
(USBRQPKTCOUNT9), offset 0x324**

**Register 309: USB Request Packet Count in Block Transfer Endpoint 10
(USBRQPKTCOUNT10), offset 0x328**

**Register 310: USB Request Packet Count in Block Transfer Endpoint 11
(USBRQPKTCOUNT11), offset 0x32C**

**Register 311: USB Request Packet Count in Block Transfer Endpoint 12
(USBRQPKTCOUNT12), offset 0x330**

**Register 312: USB Request Packet Count in Block Transfer Endpoint 13
(USBRQPKTCOUNT13), offset 0x334**

**Register 313: USB Request Packet Count in Block Transfer Endpoint 14
(USBRQPKTCOUNT14), offset 0x338**

**Register 314: USB Request Packet Count in Block Transfer Endpoint 15
(USBRQPKTCOUNT15), offset 0x33C**

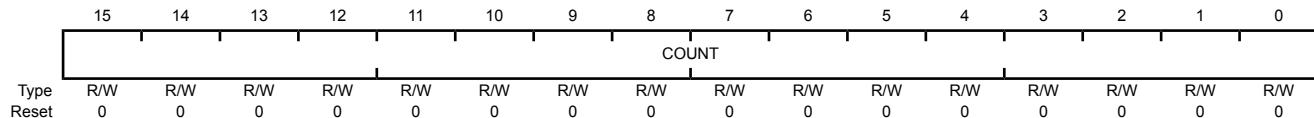
OTG A /
Host

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 USB controller uses the value recorded in this register to determine the number of requests to issue where the AUTORQ bit in the **USBRXCSRn** register has been set. See “IN Transactions as a Host” on page 923.

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	0x0000	Block Transfer Packet Count Sets the number of packets of the size defined by the MAXLOAD bit field 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 315: USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS), offset 0x340

**OTG A /
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 919).

**OTG B /
Device**

USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS)

Base 0x4005.0000

Offset 0x340

Type R/W, reset 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	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	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	EP15	R/W	0	EP15 RX Double-Packet Buffer Disable
				Value Description
			0	Disables double-packet buffering.
			1	Enables double-packet buffering.
14	EP14	R/W	0	EP14 RX Double-Packet Buffer Disable Same description as EP15.
13	EP13	R/W	0	EP13 RX Double-Packet Buffer Disable Same description as EP15.
12	EP12	R/W	0	EP12 RX Double-Packet Buffer Disable Same description as EP15.
11	EP11	R/W	0	EP11 RX Double-Packet Buffer Disable Same description as EP15.
10	EP10	R/W	0	EP10 RX Double-Packet Buffer Disable Same description as EP15.
9	EP9	R/W	0	EP9 RX Double-Packet Buffer Disable Same description as EP15.
8	EP8	R/W	0	EP8 RX Double-Packet Buffer Disable Same description as EP15.
7	EP7	R/W	0	EP7 RX Double-Packet Buffer Disable Same description as EP15.
6	EP6	R/W	0	EP6 RX Double-Packet Buffer Disable Same description as EP15.
5	EP5	R/W	0	EP5 RX Double-Packet Buffer Disable Same description as EP15.
4	EP4	R/W	0	EP4 RX Double-Packet Buffer Disable Same description as EP15.

Bit/Field	Name	Type	Reset	Description
3	EP3	R/W	0	EP3 RX Double-Packet Buffer Disable Same description as EP15.
2	EP2	R/W	0	EP2 RX Double-Packet Buffer Disable Same description as EP15.
1	EP1	R/W	0	EP1 RX Double-Packet Buffer Disable Same description as EP15.
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 316: USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS), offset 0x342

**OTG A /
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 919).

**OTG B /
Device**

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
Type	EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	reserved
Reset	R/W 0	RO 0														

Bit/Field	Name	Type	Reset	Description
15	EP15	R/W	0	EP15 TX Double-Packet Buffer Disable Value Description 0 Disables double-packet buffering. 1 Enables double-packet buffering.
14	EP14	R/W	0	EP14 TX Double-Packet Buffer Disable Same description as EP15.
13	EP13	R/W	0	EP13 TX Double-Packet Buffer Disable Same description as EP15.
12	EP12	R/W	0	EP12 TX Double-Packet Buffer Disable Same description as EP15.
11	EP11	R/W	0	EP11 TX Double-Packet Buffer Disable Same description as EP15.
10	EP10	R/W	0	EP10 TX Double-Packet Buffer Disable Same description as EP15.
9	EP9	R/W	0	EP9 TX Double-Packet Buffer Disable Same description as EP15.
8	EP8	R/W	0	EP8 TX Double-Packet Buffer Disable Same description as EP15.
7	EP7	R/W	0	EP7 TX Double-Packet Buffer Disable Same description as EP15.
6	EP6	R/W	0	EP6 TX Double-Packet Buffer Disable Same description as EP15.
5	EP5	R/W	0	EP5 TX Double-Packet Buffer Disable Same description as EP15.
4	EP4	R/W	0	EP4 TX Double-Packet Buffer Disable Same description as EP15.

Bit/Field	Name	Type	Reset	Description
3	EP3	R/W	0	EP3 TX Double-Packet Buffer Disable Same description as EP15.
2	EP2	R/W	0	EP2 TX Double-Packet Buffer Disable Same description as EP15.
1	EP1	R/W	0	EP1 TX Double-Packet Buffer Disable Same description as EP15.
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 317: USB External Power Control (USBEPC), offset 0x400

**OTG A /
Host**

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 because the fault condition may require a response faster than one provided by firmware.

**OTG B /
Device**

USB External Power Control (USBEPC)

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
reserved																
Type	RO	RO	RO	RO	RO	R/W	R/W	RO	R/W	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	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.										
9:8	PFLTACT	R/W	0x0	<p>Power Fault Action</p> <p>This bit field specifies how the USB0EPEN signal is changed when detecting a USB power fault.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Unchanged USB0EPEN is controlled by the combination of the EPEN and EPENDE bits.</td> </tr> <tr> <td>0x1</td> <td>Tristate USB0EPEN is undriven (tristate).</td> </tr> <tr> <td>0x2</td> <td>Low USB0EPEN is driven Low.</td> </tr> <tr> <td>0x3</td> <td>High USB0EPEN is driven High.</td> </tr> </tbody> </table>	Value	Description	0x0	Unchanged USB0EPEN is controlled by the combination of the EPEN and EPENDE bits.	0x1	Tristate USB0EPEN is undriven (tristate).	0x2	Low USB0EPEN is driven Low.	0x3	High USB0EPEN is driven High.
Value	Description													
0x0	Unchanged USB0EPEN is controlled by the combination of the EPEN and EPENDE bits.													
0x1	Tristate USB0EPEN is undriven (tristate).													
0x2	Low USB0EPEN is driven Low.													
0x3	High USB0EPEN is 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 This bit specifies whether a USB power fault triggers any automatic corrective action regarding the driven state of the <code>USBOEPEN</code> signal.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Disabled <code>USBOEPEN</code> is controlled by the combination of the <code>EOPEN</code> and <code>EPENDE</code> bits.</td></tr> <tr> <td>1</td><td>Enabled The <code>USBOEPEN</code> output is automatically changed to the state specified by the <code>PFLTACT</code> field.</td></tr> </tbody> </table>	Value	Description	0	Disabled <code>USBOEPEN</code> is controlled by the combination of the <code>EOPEN</code> and <code>EPENDE</code> bits.	1	Enabled The <code>USBOEPEN</code> output is automatically changed to the state specified by the <code>PFLTACT</code> field.
Value	Description									
0	Disabled <code>USBOEPEN</code> is controlled by the combination of the <code>EOPEN</code> and <code>EPENDE</code> bits.									
1	Enabled The <code>USBOEPEN</code> output is automatically changed to the state specified by the <code>PFLTACT</code> field.									
5	PFLTSEN	R/W	0	<p>Power Fault Sense This bit specifies the logical sense of the <code>USB0PFLT</code> input signal that indicates an error condition. The complementary state is the inactive state.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Low Fault If <code>USB0PFLT</code> is driven Low, the power fault is signaled internally (if enabled by the <code>PFLTEN</code> bit).</td></tr> <tr> <td>1</td><td>High Fault If <code>USB0PFLT</code> is driven High, the power fault is signaled internally (if enabled by the <code>PFLTEN</code> bit).</td></tr> </tbody> </table>	Value	Description	0	Low Fault If <code>USB0PFLT</code> is driven Low, the power fault is signaled internally (if enabled by the <code>PFLTEN</code> bit).	1	High Fault If <code>USB0PFLT</code> is driven High, the power fault is signaled internally (if enabled by the <code>PFLTEN</code> bit).
Value	Description									
0	Low Fault If <code>USB0PFLT</code> is driven Low, the power fault is signaled internally (if enabled by the <code>PFLTEN</code> bit).									
1	High Fault If <code>USB0PFLT</code> is driven High, the power fault is signaled internally (if enabled by the <code>PFLTEN</code> bit).									
4	PFLTEN	R/W	0	<p>Power Fault Input Enable This bit specifies whether the <code>USB0PFLT</code> input signal is used in internal logic.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Not Used The <code>USB0PFLT</code> signal is ignored.</td></tr> <tr> <td>1</td><td>Used The <code>USB0PFLT</code> signal is used internally.</td></tr> </tbody> </table>	Value	Description	0	Not Used The <code>USB0PFLT</code> signal is ignored.	1	Used The <code>USB0PFLT</code> signal is used internally.
Value	Description									
0	Not Used The <code>USB0PFLT</code> signal is ignored.									
1	Used The <code>USB0PFLT</code> signal is used internally.									
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						

Bit/Field	Name	Type	Reset	Description										
2	EPENDE	R/W	0	<p>EPEN Drive Enable</p> <p>This bit specifies whether the <code>USB0EPEN</code> signal is driven or undriven (tristate). When driven, the signal value is specified by the <code>EPEN</code> field. When not driven, the <code>EPEN</code> field is ignored and the <code>USB0EPEN</code> signal is placed in a high-impedance state.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Not Driven The <code>USB0EPEN</code> signal is high impedance.</td></tr> <tr> <td>1</td><td>Driven The <code>USB0EPEN</code> signal is driven to the logical value specified by the value of the <code>EPEN</code> field.</td></tr> </tbody> </table> <p>The <code>USB0EPEN</code> signal is undriven at reset because the sense of the external power supply enable is unknown. By adding the 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>	Value	Description	0	Not Driven The <code>USB0EPEN</code> signal is high impedance.	1	Driven The <code>USB0EPEN</code> signal is driven to the logical value specified by the value of the <code>EPEN</code> field.				
Value	Description													
0	Not Driven The <code>USB0EPEN</code> signal is high impedance.													
1	Driven The <code>USB0EPEN</code> signal is driven to the logical value specified by the value of the <code>EPEN</code> field.													
1:0	EPEN	R/W	0x0	<p>External Power Supply Enable Configuration</p> <p>This bit field specifies and controls the logical value driven on the <code>USB0EPEN</code> signal.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Power Enable Active Low The <code>USB0EPEN</code> signal is driven Low if the <code>EPENDE</code> bit is set.</td></tr> <tr> <td>0x1</td><td>Power Enable Active High The <code>USB0EPEN</code> signal is driven High if the <code>EPENDE</code> bit is set.</td></tr> <tr> <td>0x2</td><td>Power Enable High if VBUS Low The <code>USB0EPEN</code> signal is driven High when the A device is not recognized.</td></tr> <tr> <td>0x3</td><td>Power Enable High if VBUS High The <code>USB0EPEN</code> signal is driven High when the A device is recognized.</td></tr> </tbody> </table>	Value	Description	0x0	Power Enable Active Low The <code>USB0EPEN</code> signal is driven Low if the <code>EPENDE</code> bit is set.	0x1	Power Enable Active High The <code>USB0EPEN</code> signal is driven High if the <code>EPENDE</code> bit is set.	0x2	Power Enable High if VBUS Low The <code>USB0EPEN</code> signal is driven High when the A device is not recognized.	0x3	Power Enable High if VBUS High The <code>USB0EPEN</code> signal is driven High when the A device is recognized.
Value	Description													
0x0	Power Enable Active Low The <code>USB0EPEN</code> signal is driven Low if the <code>EPENDE</code> bit is set.													
0x1	Power Enable Active High The <code>USB0EPEN</code> signal is driven High if the <code>EPENDE</code> bit is set.													
0x2	Power Enable High if VBUS Low The <code>USB0EPEN</code> signal is driven High when the A device is not recognized.													
0x3	Power Enable High if VBUS High The <code>USB0EPEN</code> signal is driven High when the A device is recognized.													

Register 318: USB External Power Control Raw Interrupt Status (USBEPCRIS), offset 0x404

**OTG A /
Host**

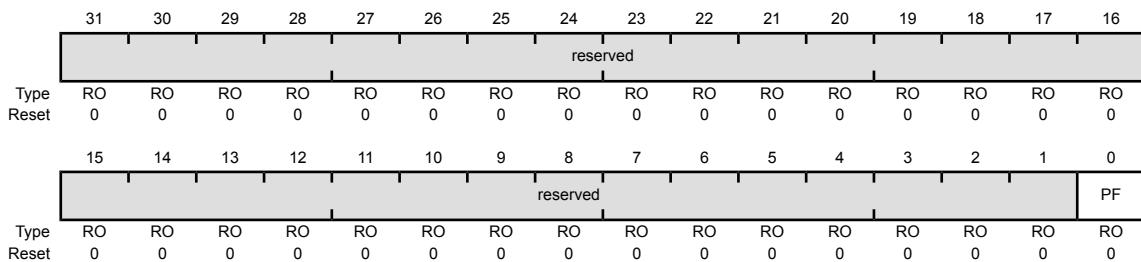
This 32-bit register specifies the unmasked interrupt status of the two-pin external power interface.

USB External Power Control Raw Interrupt Status (USBEPCRIS)

Base 0x4005.0000

Offset 0x404

Type RO, reset 0x0000.0000



This bit is cleared by writing a 1 to the **PF** bit in the **USBEPCISC** register.

Register 319: USB External Power Control Interrupt Mask (USBEPCIM), offset 0x408

**OTG A /
Host**

This 32-bit register specifies the interrupt mask of the two-pin external power interface.

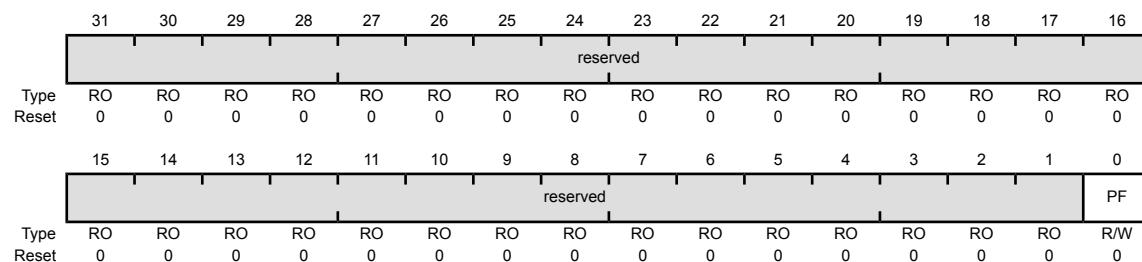
USB External Power Control Interrupt Mask (USBEPCIM)

Base 0x4005.0000

Offset 0x408

Type R/W, reset 0x0000.0000

**OTG B /
Device**



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	PF	R/W	0	USB Power Fault Interrupt Mask Value Description 1 The raw interrupt signal from a detected power fault is sent to the interrupt controller. 0 A detected power fault does not affect the interrupt status.

Register 320: USB External Power Control Interrupt Status and Clear (USBEPCISC), offset 0x40C

**OTG A /
Host**

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.

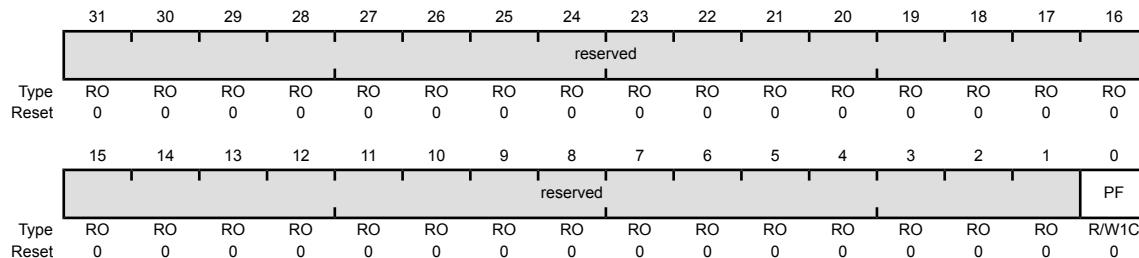
**OTG B /
Device**

USB External Power Control Interrupt Status and Clear (USBEPCISC)

Base 0x4005.0000

Offset 0x40C

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	PF	R/W1C	0	USB Power Fault Interrupt Status and Clear
	Value	Description		
	1	The PF bits in the USBEPCRIS and USBEPCIM registers are set, providing an interrupt to the interrupt controller.		
	0	No interrupt has occurred or the interrupt is masked.		
	This bit is cleared by writing a 1. Clearing this bit also clears the PF bit in the USBEPCRIS register.			

Register 321: USB Device RESUME Raw Interrupt Status (USBDRRIS), offset 0x410

**OTG A /
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.

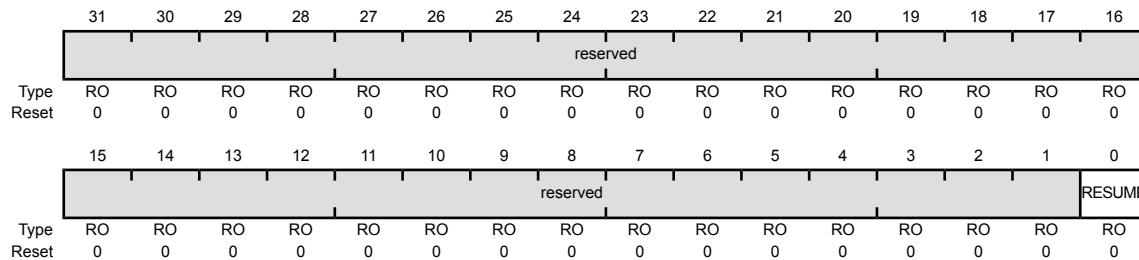
USB Device RESUME Raw Interrupt Status (USBDRRIS)

Base 0x4005.0000

Offset 0x410

Type RO, reset 0x0000.0000

**OTG B /
Device**



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	RESUME	RO	0	RESUME Interrupt Status
		Value	Description	
		1	A RESUME status has been detected.	
		0	An interrupt has not occurred.	
		This bit is cleared by writing a 1 to the RESUME bit in the USBDRISC register.		

Register 322: USB Device RESUME Interrupt Mask (USBDRIM), offset 0x414

**OTG A /
Host**

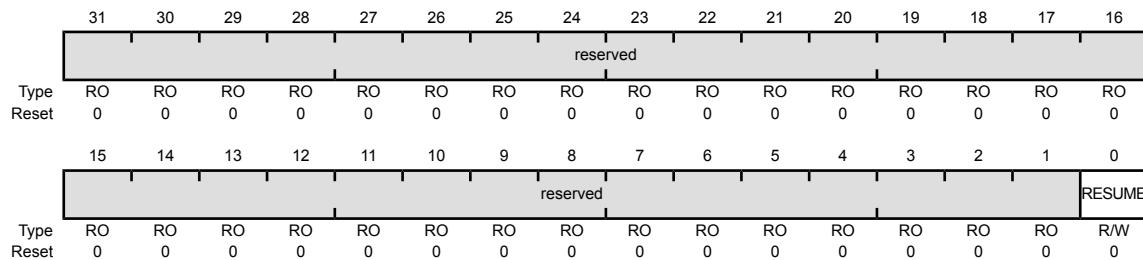
The **USBDRIM** 32-bit register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

USB Device RESUME Interrupt Mask (USBDRIM)

Base 0x4005.0000

Offset 0x414

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	RESUME	R/W	0	RESUME Interrupt Mask
				Value Description
			1	The raw interrupt signal from a detected RESUME is sent to the interrupt controller. This bit should only be set when a SUSPEND has been detected (the SUSPEND bit in the USBIS register is set).
			0	A detected RESUME does not affect the interrupt status.

Register 323: USB Device RESUME Interrupt Status and Clear (USBDRISC), offset 0x418

**OTG A /
Host**

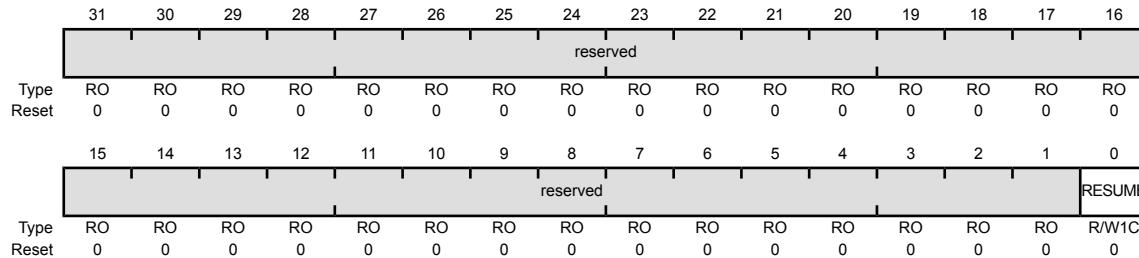
The **USBDRISC** 32-bit register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

USB Device RESUME Interrupt Status and Clear (USBDRISC)

Base 0x4005.0000

Offset 0x418

Type W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	RESUME	R/W1C	0	RESUME Interrupt Status and Clear
	Value	Description		
	1	The RESUME bits in the USBDRRIS and USBDRCIM registers are set, providing an interrupt to the interrupt controller.		
	0	No interrupt has occurred or the interrupt is masked.		
	This bit is cleared by writing a 1. Clearing this bit also clears the RESUME bit in the USBDRCRIS register.			

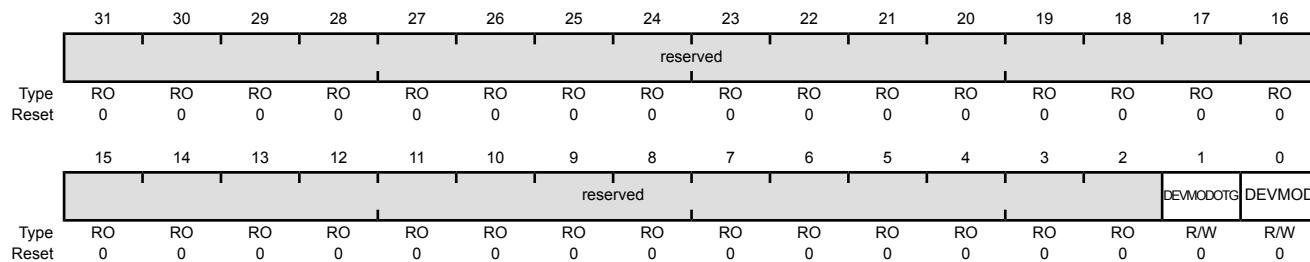
Register 324: USB General-Purpose Control and Status (USBGPCS), offset 0x41C

OTG A / Host

Note: When used in OTG mode, `USB0VBUS` and `USB0ID` do not require any configuration as they are dedicated pins for the USB controller and directly connect to the USB connector's VBUS and ID signals. If the USB controller is used as either a dedicated Host or Device, the `DEVMODOTG` and `DEVMOD` bits in the **USB General-Purpose Control and Status (USBGPCS)** register can be used to connect the `USB0VBUS` and `USB0ID` inputs to fixed levels internally, freeing the `PB0` and `PB1` pins for GPIO use. For proper self-powered Device operation, the VBUS value must still be monitored to assure that if the Host removes VBUS, the self-powered Device disables the D+/D- pull-up resistors. This function can be accomplished by connecting a standard GPIO to VBUS.

USB General-Purpose Control and Status (USBGPCS)

Base 0x4005.0000
Offset 0x41C
Type R/W, reset 0x0000.0000



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	DEVMODOTG	R/W	0	<p>Enable Device Mode</p> <p>This bit enables the <code>DEVMOD</code> bit to control the state of the internal ID signal in OTG mode.</p> <table border="0"> <tr> <td>Value</td><td>Description</td></tr> <tr> <td>0</td><td>The mode is specified by the state of the internal ID signal.</td></tr> <tr> <td>1</td><td>This bit enables the <code>DEVMOD</code> bit to control the internal ID signal.</td></tr> </table>	Value	Description	0	The mode is specified by the state of the internal ID signal.	1	This bit enables the <code>DEVMOD</code> bit to control the internal ID signal.
Value	Description									
0	The mode is specified by the state of the internal ID signal.									
1	This bit enables the <code>DEVMOD</code> bit to control the internal ID signal.									
0	DEVMOD	R/W	0	<p>Device Mode</p> <p>This bit specifies the state of the internal ID signal in Host mode and in OTG mode when the <code>DEVMODOTG</code> bit is set.</p> <p>In Device mode this bit is ignored (assumed set).</p> <table border="0"> <tr> <td>Value</td><td>Description</td></tr> <tr> <td>0</td><td>Host mode</td></tr> <tr> <td>1</td><td>Device mode</td></tr> </table>	Value	Description	0	Host mode	1	Device mode
Value	Description									
0	Host mode									
1	Device mode									

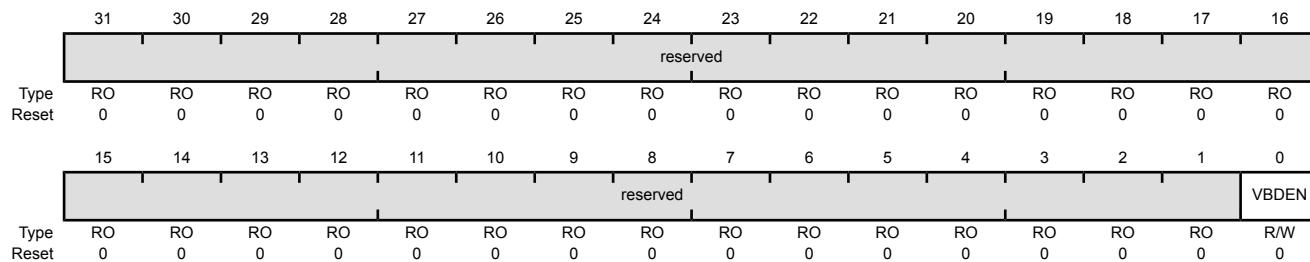
Register 325: USB VBUS Droop Control (USBVDC), offset 0x430

**OTG A /
Host**

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 VBUS Valid level (4.75 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.

USB VBUS Droop Control (USBVDC)

Base 0x4005.0000
Offset 0x430
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	VBDEN	R/W	0	VBUS Droop Enable
		Value	Description	
		0	No effect.	
		1	Any changes from VBUSVALID are masked when VBUS goes below 4.75 V but not lower than 2.0 V for 65 microseconds. During this time, the VBUS state indicates VBUSVALID.	

Register 326: USB VBUS Droop Control Raw Interrupt Status (USBVDCRIS), offset 0x434

**OTG A /
Host**

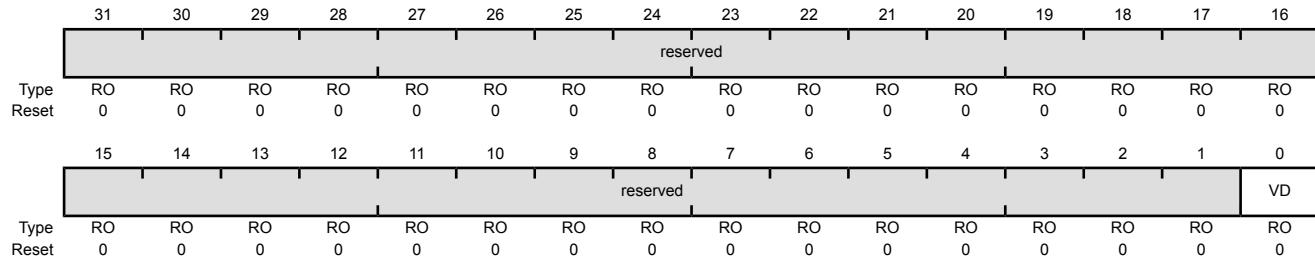
This 32-bit register specifies the unmasked interrupt status of the VBUS droop limit of 65 microseconds.

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 0x0000.0000 Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

0 VD RO 0 VBUS Droop Raw Interrupt Status

Value Description

1 A VBUS droop lasting for 65 microseconds has been detected.
0 An interrupt has not occurred.

This bit is cleared by writing a 1 to the VD bit in the **USBVDCISC** register.

Register 327: USB VBUS Droop Control Interrupt Mask (USBVDCIM), offset 0x438

OTG A / Host	This 32-bit register specifies the interrupt mask of the VBUS droop.																															
USB VBUS Droop Control Interrupt Mask (USBVDCIM)																																
Base 0x4005.0000 Offset 0x438																																
Type R/W, reset 0x0000.0000																																

Bit/Field	Name	Type	Reset	Description						
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
0	VD	R/W	0	VBUS Droop Interrupt Mask <table style="margin-left: 20px; border-collapse: collapse;"> <tr> <th style="text-align: left;">Value</th> <th style="text-align: left;">Description</th> </tr> <tr> <td style="text-align: left;">1</td> <td>The raw interrupt signal from a detected VBUS droop is sent to the interrupt controller.</td> </tr> <tr> <td style="text-align: left;">0</td> <td>A detected VBUS droop does not affect the interrupt status.</td> </tr> </table>	Value	Description	1	The raw interrupt signal from a detected VBUS droop is sent to the interrupt controller.	0	A detected VBUS droop does not affect the interrupt status.
Value	Description									
1	The raw interrupt signal from a detected VBUS droop is sent to the interrupt controller.									
0	A detected VBUS droop does not affect the interrupt status.									

Register 328: USB VBUS Droop Control Interrupt Status and Clear (USBVDCISC), offset 0x43C

**OTG A /
Host**

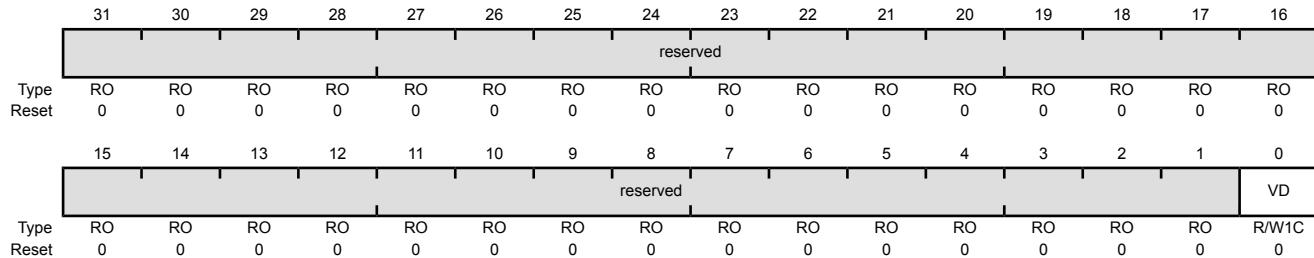
This 32-bit register specifies the masked interrupt status of the VBUS droop and provides a method to clear the interrupt state.

USB VBUS Droop Control Interrupt Status and Clear (USBVDCISC)

Base 0x4005.0000

Offset 0x43C

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	VD	R/W1C	0	VBUS Droop Interrupt Status and Clear
	Value	Description		
	1	The VD bits in the USBVDCRIS and USBVDCIM registers are set, providing an interrupt to the interrupt controller.		
	0	No interrupt has occurred or the interrupt is masked.		

This bit is cleared by writing a 1. Clearing this bit also clears the **VD** bit in the **USBVDCRIS** register.

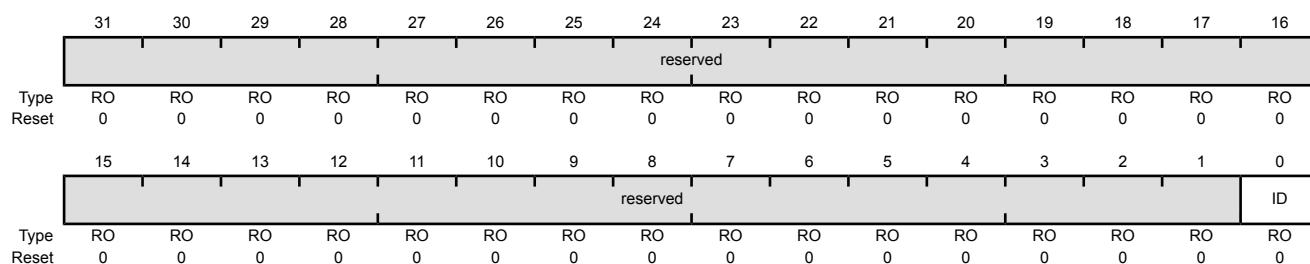
Register 329: USB ID Valid Detect Raw Interrupt Status (USBIDVRIS), offset 0x444

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



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	ID	RO	0	ID Valid Detect Raw Interrupt Status

Value	Description
1	A valid ID has been detected.
0	An interrupt has not occurred.

This bit is cleared by writing a 1 to the **ID** bit in the **USBIDVISC** register.

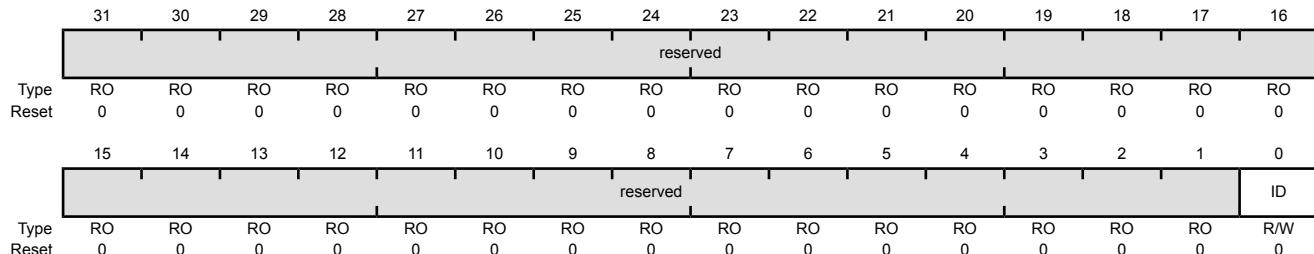
Register 330: USB ID Valid Detect Interrupt Mask (USBIDVIM), offset 0x448

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	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	ID	R/W	0	ID Valid Detect Interrupt Mask
	Value	Description		
	1	The raw interrupt signal from a detected ID valid is sent to the interrupt controller.		
	0	A detected ID valid does not affect the interrupt status.		

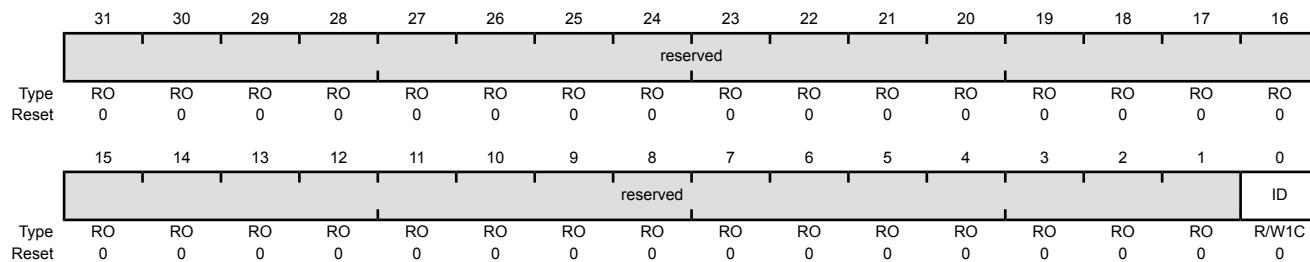
Register 331: USB ID Valid Detect Interrupt Status and Clear (USPIDVISC), offset 0x44C

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 (USPIDVISC)

Base 0x4005.0000
Offset 0x44C
Type R/W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	ID	R/W1C	0	ID Valid Detect Interrupt Status and Clear
	Value	Description		
	1	The ID bits in the USBIDVRIS and USBIDVIM registers are set, providing an interrupt to the interrupt controller.		
	0	No interrupt has occurred or the interrupt is masked.		
	This bit is cleared by writing a 1. Clearing this bit also clears the ID bit in the USBIDVRIS register.			

Register 332: USB DMA Select (USBDMASEL), offset 0x450

**OTG A /
Host**

This 32-bit register specifies which endpoints are mapped to the 6 allocated µDMA channels, see Table 8-1 on page 367 for more information on channel assignments.

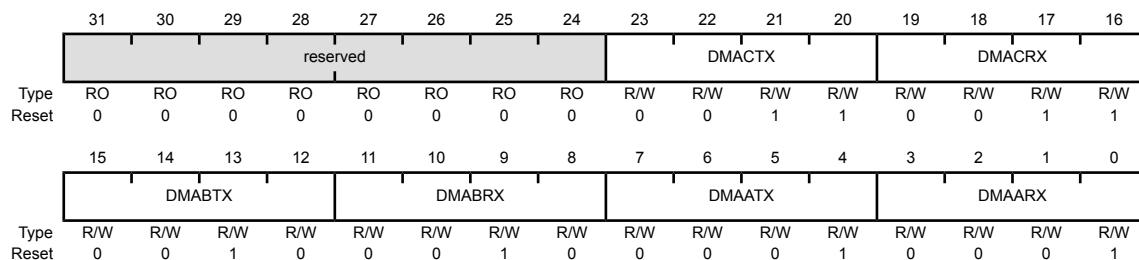
USB DMA Select (USBDMASEL)

Base 0x4005.0000

Offset 0x450

Type R/W, reset 0x0033.2211

**OTG B /
Device**



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.

23:20 DMACTX R/W 0x3 DMA C TX Select
Specifies the TX mapping of the third USB endpoint on µDMA channel 5 (primary assignment).

Value	Description
0x0	reserved
0x1	Endpoint 1 TX
0x2	Endpoint 2 TX
0x3	Endpoint 3 TX
0x4	Endpoint 4 TX
0x5	Endpoint 5 TX
0x6	Endpoint 6 TX
0x7	Endpoint 7 TX
0x8	Endpoint 8 TX
0x9	Endpoint 9 TX
0xA	Endpoint 10 TX
0xB	Endpoint 11 TX
0xC	Endpoint 12 TX
0xD	Endpoint 13 TX
0xE	Endpoint 14 TX
0xF	Endpoint 15 TX

Bit/Field	Name	Type	Reset	Description
19:16	DMACRX	R/W	0x3	DMA C RX Select Specifies the RX and TX mapping of the third USB endpoint on µDMA channel 4 (primary assignment).
				Value Description
			0x0	reserved
			0x1	Endpoint 1 RX
			0x2	Endpoint 2 RX
			0x3	Endpoint 3 RX
			0x4	Endpoint 4 RX
			0x5	Endpoint 5 RX
			0x6	Endpoint 6 RX
			0x7	Endpoint 7 RX
			0x8	Endpoint 8 RX
			0x9	Endpoint 9 RX
			0xA	Endpoint 10 RX
			0xB	Endpoint 11 RX
			0xC	Endpoint 12 RX
			0xD	Endpoint 13 RX
			0xE	Endpoint 14 RX
			0xF	Endpoint 15 RX
15:12	DMABTX	R/W	0x2	DMA B TX Select Specifies the TX mapping of the second USB endpoint on µDMA channel 3 (primary assignment). Same bit definitions as the DMACTX field.
11:8	DMABRX	R/W	0x2	DMA B RX Select Specifies the RX mapping of the second USB endpoint on µDMA channel 2 (primary assignment). Same bit definitions as the DMACRX field.
7:4	DMAATX	R/W	0x1	DMA A TX Select Specifies the TX mapping of the first USB endpoint on µDMA channel 1 (primary assignment). Same bit definitions as the DMACTX field.
3:0	DMAARX	R/W	0x1	DMA A RX Select Specifies the RX mapping of the first USB endpoint on µDMA channel 0 (primary assignment). Same bit definitions as the DMACRX 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. See “Signal Description” on page 1054 for more information.

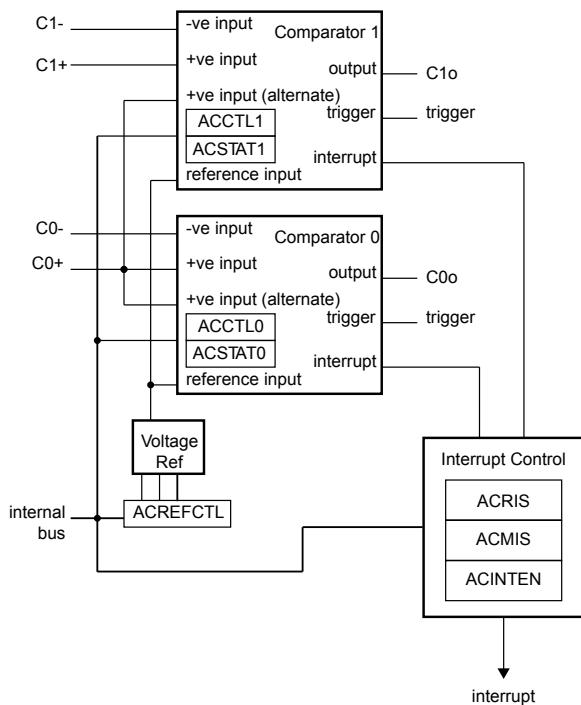
The comparator can provide its output to a device pin, acting as a replacement for an analog comparator on the board. In addition, the comparator can signal the application via interrupts or trigger the start of a sample sequence in the ADC. The interrupt generation and ADC triggering logic is separate and independent. This flexibility means, for example, that an interrupt can be generated on a rising edge and the ADC triggered on a falling edge.

The Stellaris® LM3S9L97 microcontroller provides two independent integrated analog comparators with 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 the following 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 Signal Description

Table 20-1 on page 1054 and Table 20-2 on page 1054 list the external signals of the Analog Comparators and describe the function of each. The Analog Comparator output signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the Analog Comparator signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 447) should be set to choose the Analog Comparator function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 465) to assign the Analog Comparator signal to the specified GPIO port pin. The positive and negative input signals are configured by clearing the DEN bit in the **GPIO Digital Enable (GPIODEN)** register. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 423.

Table 20-1. Signals for Analog Comparators (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
C0+	90	PB6	I	Analog	Analog comparator 0 positive input.
C0-	92	PB4	I	Analog	Analog comparator 0 negative input.
C0o	24 42 90 91 100	PC5 (3) PF4 (2) PB6 (3) PB5 (1) PD7 (2)	O	TTL	Analog comparator 0 output.
C1+	24	PC5	I	Analog	Analog comparator 1 positive input.
C1-	91	PB5	I	Analog	Analog comparator 1 negative input.
C1o	2 22 24 41 84	PE6 (2) PC7 (7) PC5 (2) PF5 (2) PH2 (2)	O	TTL	Analog comparator 1 output.

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

Table 20-2. Signals for Analog Comparators (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
C0+	A7	PB6	I	Analog	Analog comparator 0 positive input.
C0-	A6	PB4	I	Analog	Analog comparator 0 negative input.
C0o	M1 K4 A7 B7 A2	PC5 (3) PF4 (2) PB6 (3) PB5 (1) PD7 (2)	O	TTL	Analog comparator 0 output.
C1+	M1	PC5	I	Analog	Analog comparator 1 positive input.
C1-	B7	PB5	I	Analog	Analog comparator 1 negative input.
C1o	A1 L2 M1 K3 D11	PE6 (2) PC7 (7) PC5 (2) PF5 (2) PH2 (2)	O	TTL	Analog comparator 1 output.

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

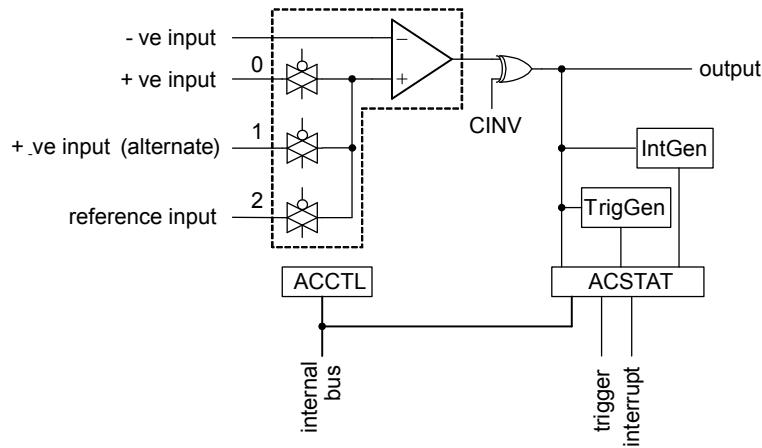
20.3 Functional Description

The comparator compares the VIN- and VIN+ inputs to produce an output, VOUT.

$$\begin{aligned} \text{VIN-} < \text{VIN+}, \text{ VOUT} &= 1 \\ \text{VIN-} > \text{VIN+}, \text{ VOUT} &= 0 \end{aligned}$$

As shown in Figure 20-2 on page 1055, the input source for VIN- is an external input, Cn-. In addition to an external input, Cn+, input sources for VIN+ can be the C0+ or an internal reference, V_{REF}.

Figure 20-2. Structure of Comparator Unit



A comparator is configured through two status/control registers, **Analog Comparator Control (ACCTL)** and **Analog Comparator Status (ACSTAT)**. The internal reference is configured through one control register, **Analog Comparator Reference Voltage Control (ACREFCTL)**. Interrupt status and control are configured through three registers, **Analog Comparator Masked Interrupt Status (ACMIS)**, **Analog Comparator Raw Interrupt Status (ACRIS)**, and **Analog Comparator Interrupt Enable (ACINTEN)**.

Typically, the comparator output is used internally to generate an interrupt as controlled by the ISEN bit in the **ACCTL** register. The output may also be used to drive an external pin, Co or generate an analog-to-digital converter (ADC) trigger.

Important: The ASRCP bits in the **ACCTL** register must be set before using the analog comparators.

20.3.1 Internal Reference Programming

The structure of the internal reference is shown in Figure 20-3 on page 1056. The internal reference is controlled by a single configuration register (**ACREFCTL**). Table 20-3 on page 1056 shows the programming options to develop specific internal reference values, to compare an external voltage against a particular voltage generated internally (V_{REF}).

Figure 20-3. Comparator Internal Reference Structure

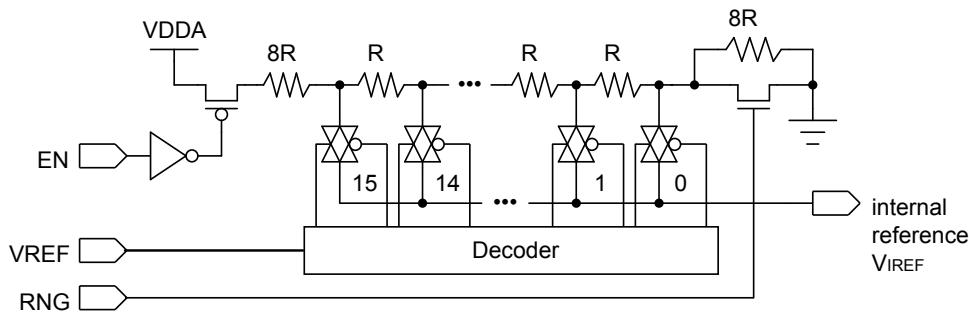


Table 20-3. Internal Reference Voltage and ACREFCTL Field Values

ACREFCTL Register		Output Reference Voltage Based on VREF Field Value
EN Bit Value	RNG Bit Value	
EN=0	RNG=X	0 V (GND) for any value of V_{REF} ; however, it is recommended that $RNG=1$ and $V_{REF}=0$ for the least noisy ground reference.
	RNG=0	<p>Total resistance in ladder is 31 R.</p> $V_{IREF} = V_{DDA} \times \frac{R_{VREF}}{R_T}$ $V_{IREF} = V_{DDA} \times \frac{(V_{REF} + 8)}{31}$ $V_{IREF} = 0.85 + 0.106 \times V_{REF}$ <p>The range of internal reference in this mode is 0.85-2.448 V.</p>
EN=1	RNG=1	<p>Total resistance in ladder is 23 R.</p> $V_{IREF} = V_{DDA} \times \frac{R_{VREF}}{R_T}$ $V_{IREF} = V_{DDA} \times \frac{V_{REF}}{23}$ $V_{IREF} = 0.143 \times V_{REF}$ <p>The range of internal reference for this mode is 0-2.152 V.</p>

20.4 Initialization and Configuration

The following example shows how to configure an analog comparator to read back its output value from an internal register.

1. Enable the analog comparator 0 clock by writing a value of 0x0010.0000 to the **RCGC1** register in the System Control module (see page 276).
2. In the GPIO module, enable the GPIO port/pin associated with the input signals as GPIO inputs. To determine which GPIO to configure, see Table 24-4 on page 1182.
3. Configure the **PMCn** fields in the **GPIOPCTL** register to assign the analog comparator output signals to the appropriate pins (see page 465 and Table 24-5 on page 1190).
4. Configure the internal voltage reference to 1.65 V by writing the **ACREFCTL** register with the value 0x0000.030C.
5. Configure the comparator to use the internal voltage reference and to *not* invert the output by writing the **ACCTLn** register with the value of 0x0000.040C.
6. Delay for 10 μ s.
7. Read the comparator output value by reading the **ACSTATn** register's OVAL value.

Change the level of the comparator negative input signal C- to see the OVAL value change.

20.5 Register Map

Table 20-4 on page 1057 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 276).

Table 20-4. Analog Comparators Register Map

Offset	Name	Type	Reset	Description	See page
0x000	ACMIS	R/W1C	0x0000.0000	Analog Comparator Masked Interrupt Status	1058
0x004	ACRIS	RO	0x0000.0000	Analog Comparator Raw Interrupt Status	1059
0x008	ACINTEN	R/W	0x0000.0000	Analog Comparator Interrupt Enable	1060
0x010	ACREFCTL	R/W	0x0000.0000	Analog Comparator Reference Voltage Control	1061
0x020	ACSTAT0	RO	0x0000.0000	Analog Comparator Status 0	1062
0x024	ACCTL0	R/W	0x0000.0000	Analog Comparator Control 0	1063
0x040	ACSTAT1	RO	0x0000.0000	Analog Comparator Status 1	1062
0x044	ACCTL1	R/W	0x0000.0000	Analog Comparator Control 1	1063

20.6 Register Descriptions

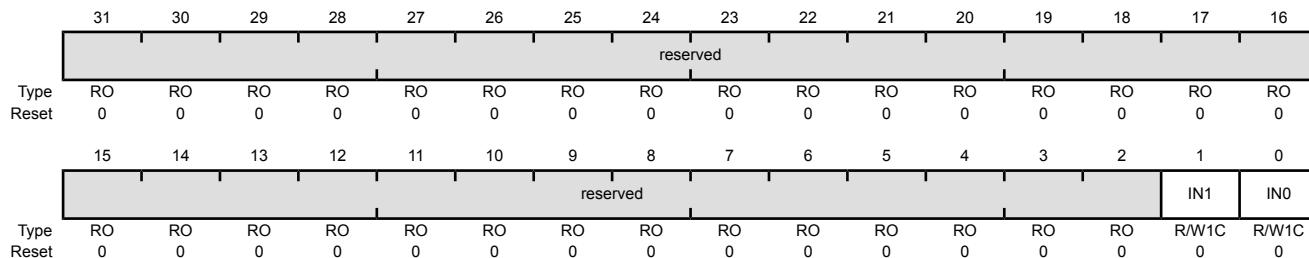
The remainder of this section lists and describes the Analog Comparator registers, in numerical order by address offset.

Register 1: Analog Comparator Masked Interrupt Status (ACMIS), offset 0x000

This register provides a summary of the interrupt status (masked) of the comparators.

Analog Comparator Masked Interrupt Status (ACMIS)

Base 0x4003.C000
Offset 0x000
Type R/W1C, reset 0x0000.0000



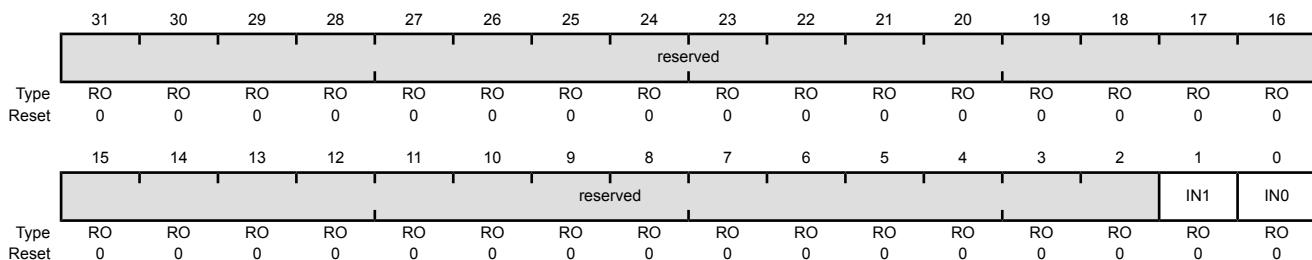
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	IN1	R/W1C	0	Comparator 1 Masked Interrupt Status
	Value Description			
	1	The IN1 bits in the ACRIS register and the ACINTEN registers are set, providing an interrupt to the interrupt controller.		
	0	No interrupt has occurred or the interrupt is masked.		
	This bit is cleared by writing a 1. Clearing this bit also clears the IN1 bit in the ACRIS register.			
0	IN0	R/W1C	0	Comparator 0 Masked Interrupt Status
	Value Description			
	1	The IN0 bits in the ACRIS register and the ACINTEN registers are set, providing an interrupt to the interrupt controller.		
	0	No interrupt has occurred or the interrupt is masked.		
	This bit is cleared by writing a 1. Clearing this bit also clears the IN0 bit in the ACRIS register.			

Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004

This register provides a summary of the interrupt status (raw) of the comparators. The bits in this register must be enabled to generate interrupts using the **ACINTEN** register.

Analog Comparator Raw Interrupt Status (ACRIS)

Base 0x4003.C000
Offset 0x004
Type RO, reset 0x0000.0000



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	IN1	RO	0	Comparator 1 Interrupt Status
0	IN0	RO	0	Comparator 0 Interrupt Status

Value Description
1 Comparator 1 has generated an interrupt for an event as configured by the ISEN bit in the **ACCTL1** register.
0 An interrupt has not occurred.

This bit is cleared by writing a 1 to the IN1 bit in the **ACMIS** register.

Value Description
1 Comparator 0 has generated an interrupt for an event as configured by the ISEN bit in the **ACCTL0** register.
0 An interrupt has not occurred.

This bit is cleared by writing a 1 to the IN0 bit in the **ACMIS** register.

Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x008

This register provides the interrupt enable for the comparators.

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													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO	R/W	R/W	R/W												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

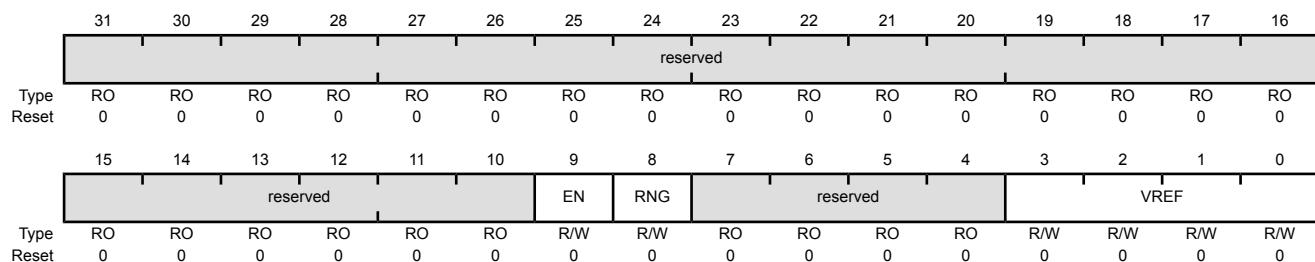
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	IN1	R/W	0	Comparator 1 Interrupt Enable
		Value	Description	
		1	The raw interrupt signal comparator 1 is sent to the interrupt controller.	
		0	A comparator 1 interrupt does not affect the interrupt status.	
0	IN0	R/W	0	Comparator 0 Interrupt Enable
		Value	Description	
		1	The raw interrupt signal comparator 0 is sent to the interrupt controller.	
		0	A comparator 0 interrupt does not affect the interrupt status.	

Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x010

This register specifies whether the resistor ladder is powered on as well as the range and tap.

Analog Comparator Reference Voltage Control (ACREFCTL)

Base 0x4003.C000
Offset 0x010
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	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	EN	R/W	0	<p>Resistor Ladder Enable</p> <p>Value Description</p> <p>0 The resistor ladder is unpowered.</p> <p>1 Powers on the resistor ladder. The resistor ladder is connected to V_{DDA}.</p> <p>This bit is cleared at reset so that the internal reference consumes the least amount of power if it is not used.</p>
8	RNG	R/W	0	<p>Resistor Ladder Range</p> <p>Value Description</p> <p>0 The resistor ladder has a total resistance of 31 R.</p> <p>1 The resistor ladder has a total resistance of 23 R.</p>
7:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	VREF	R/W	0x0	<p>Resistor Ladder Voltage Ref</p> <p>The VREF bit field specifies the resistor ladder tap that is passed through an analog multiplexer. The voltage corresponding to the tap position is the internal reference voltage available for comparison. See Table 20-3 on page 1056 for some output reference voltage examples.</p>

Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x020**Register 6: Analog Comparator Status 1 (ACSTAT1), offset 0x040**

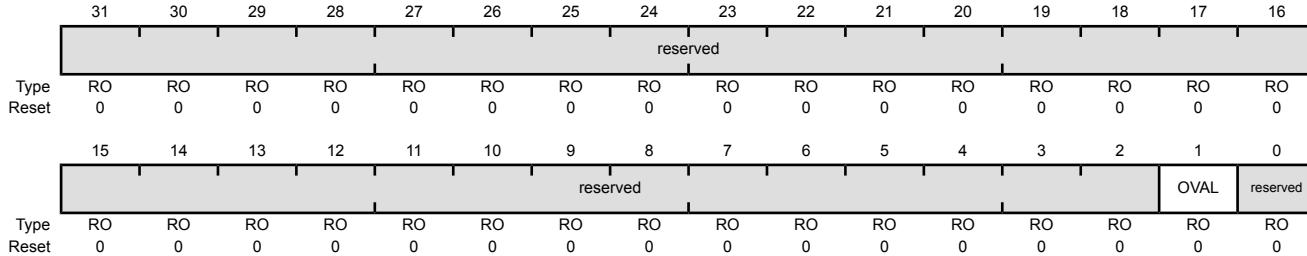
These registers specify the current output value of the comparator.

Analog Comparator Status 0 (ACSTAT0)

Base 0x4003.C000

Offset 0x020

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	OVAL	RO	0	Comparator Output Value
		Value	Description	
	0	VIN- > VIN+		
	1	VIN- < VIN+		
		VIN -	is the voltage on the Cn- pin. VIN+ is the voltage on the Cn+ pin, the C0+ pin, or the internal voltage reference (V_{IREF}) as defined by the ASRCP bit in the ACCTL register.	
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: Analog Comparator Control 0 (ACCTL0), offset 0x024**Register 8: Analog Comparator Control 1 (ACCTL1), offset 0x044**

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				TOEN	ASRCP	reserved	TSLVAL	TSEN	ISLVAL	ISEN	CINV	reserved			
Type	RO	RO	RO	RO	R/W	R/W	RO	R/W	R/W	R/W	R/W	R/W	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
31:12	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.
11	TOEN	R/W	0	Trigger Output Enable
		Value	Description	
		0	ADC events are suppressed and not sent to the ADC.	
		1	ADC events are sent to the ADC.	
10:9	ASRCP	R/W	0x0	Analog Source Positive The ASRCP field specifies the source of input voltage to the VIN+ terminal of the comparator. The encodings for this field are as follows:
		Value	Description	
		0x0	Pin value of Cn+	
		0x1	Pin value of C0+	
		0x2	Internal voltage reference (V _{REF})	
		0x3	Reserved	
8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TSLVAL	R/W	0	Trigger Sense Level Value
		Value	Description	
		0	An ADC event is generated if the comparator output is Low.	
		1	An ADC event is generated if the comparator output is High.	

Bit/Field	Name	Type	Reset	Description										
6:5	TSEN	R/W	0x0	<p>Trigger Sense</p> <p>The TSEN field specifies the sense of the comparator output that generates an ADC event. The sense conditioning is as follows:</p> <table> <thead> <tr> <th>Value</th><th>Description</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	Description	0x0	Level sense, see ISLVAL	0x1	Falling edge	0x2	Rising edge	0x3	Either edge
Value	Description													
0x0	Level sense, see ISLVAL													
0x1	Falling edge													
0x2	Rising edge													
0x3	Either edge													
4	ISLVAL	R/W	0	<p>Interrupt Sense Level Value</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>An interrupt is generated if the comparator output is Low.</td></tr> <tr> <td>1</td><td>An interrupt is generated if the comparator output is High.</td></tr> </tbody> </table>	Value	Description	0	An interrupt is generated if the comparator output is Low.	1	An interrupt is generated if the comparator output is High.				
Value	Description													
0	An interrupt is generated if the comparator output is Low.													
1	An interrupt is generated if the comparator output is High.													
3:2	ISEN	R/W	0x0	<p>Interrupt Sense</p> <p>The ISEN field specifies the sense of the comparator output that generates an interrupt. The sense conditioning is as follows:</p> <table> <thead> <tr> <th>Value</th><th>Description</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	Description	0x0	Level sense, see ISLVAL	0x1	Falling edge	0x2	Rising edge	0x3	Either edge
Value	Description													
0x0	Level sense, see ISLVAL													
0x1	Falling edge													
0x2	Rising edge													
0x3	Either edge													
1	CINV	R/W	0	<p>Comparator Output Invert</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The output of the comparator is unchanged.</td></tr> <tr> <td>1</td><td>The output of the comparator is inverted prior to being processed by hardware.</td></tr> </tbody> </table>	Value	Description	0	The output of the comparator is unchanged.	1	The output of the comparator is inverted prior to being processed by hardware.				
Value	Description													
0	The output of the comparator is unchanged.													
1	The output of the comparator is inverted prior to being processed by hardware.													
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.										

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® PWM module consists of three PWM generator blocks and a control block. The control block determines the polarity of the PWM signals, and which signals are passed through to the pins.

Each PWM generator block produces two PWM signals that share the same timer and frequency and can either be programmed with independent actions or as a single pair of complementary signals with dead-band delays inserted. The output signals, pwmA' and pwmb', of the PWM generation blocks are managed by the output control block before being passed to the device pins as PWM0 and PWM1 or PWM2 and PWM3, and so on.

The Stellaris PWM module provides a great deal of flexibility and can generate simple PWM signals, such as those required by a simple charge pump as well as 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.

The Stellaris LM3S9L97 PWM module consists of three PWM generator blocks and a control block. Each PWM generator block has the following features:

- Four fault-condition handling input to quickly provide low-latency shutdown and prevent damage to the motor being controlled
- 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

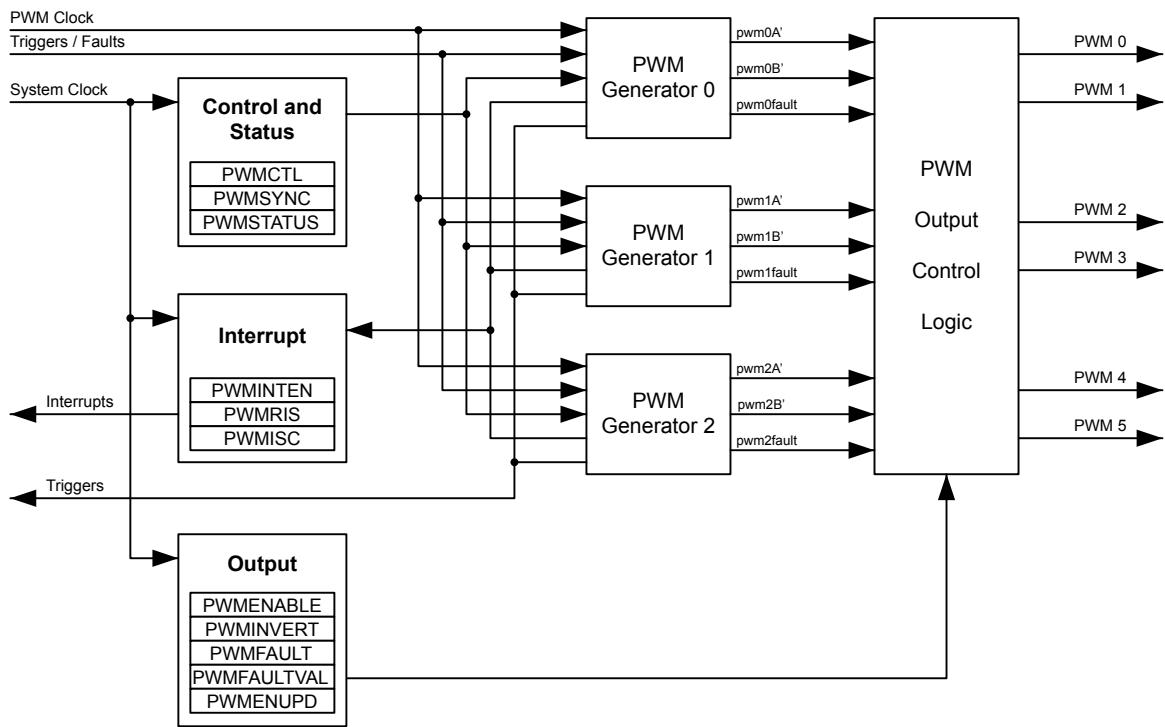
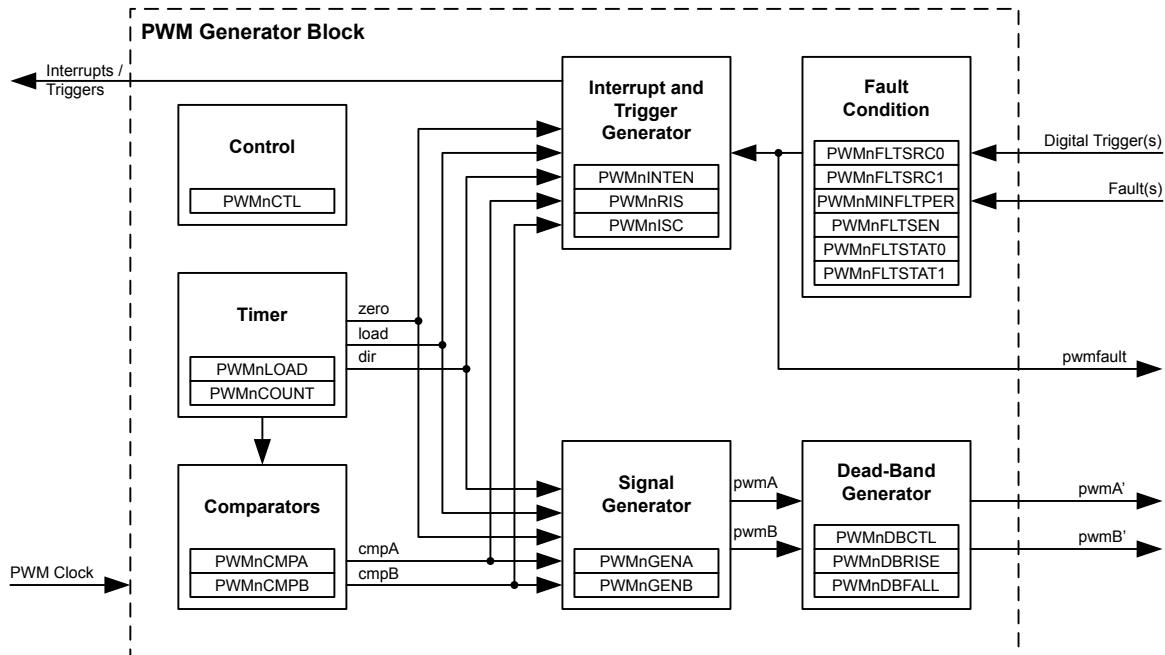
- Can initiate an ADC sample sequence

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
- Synchronization of PWM output enables 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 1067 provides the Stellaris PWM module unit diagram and Figure 21-2 on page 1067 provides a more detailed diagram of a Stellaris PWM generator. The LM3S9L97 controller contains three generator blocks (PWM0, PWM1, and PWM2) and generates six independent PWM signals or three paired PWM signals with dead-band delays inserted.

Figure 21-1. PWM Unit Diagram**Figure 21-2. PWM Module Block Diagram**

21.2 Signal Description

Table 21-1 on page 1068 and Table 21-2 on page 1069 list the external signals of the PWM module and describe the function of each. The PWM controller signals are alternate functions for some GPIO

signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these PWM signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 447) should be set to choose the PWM function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 465) to assign the PWM signal to the specified GPIO port pin. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 423.

Table 21-1. Signals for PWM (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
Fault0	6 39 42 65 75 83 99	PE4 (4) PJ2 (10) PF4 (4) PB3 (2) PE1 (3) PH3 (2) PD6 (1)	I	TTL	PWM Fault 0.
Fault1	90	PB6 (4)	I	TTL	PWM Fault 1.
Fault2	24 63	PC5 (4) PH5 (10)	I	TTL	PWM Fault 2.
Fault3	65 84	PB3 (4) PH2 (4)	I	TTL	PWM Fault 3.
PWM0	10 14 19 34 47	PD0 (1) PJ0 (10) PG0 (2) PA6 (4) PF0 (3)	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
PWM1	11 18 35 61 87	PD1 (1) PG1 (2) PA7 (4) PF1 (3) PJ1 (10)	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
PWM2	12 60 66 86	PD2 (3) PF2 (4) PB0 (2) PH0 (2)	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
PWM3	13 59 67 85	PD3 (3) PF3 (4) PB1 (2) PH1 (2)	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
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. This signal is controlled by PWM Generator 2.

Table 21-1. Signals for PWM (100LQFP) (continued)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
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. This signal is controlled by PWM Generator 2.

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

Table 21-2. Signals for PWM (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
Fault0	B2 K6 K4 E11 A12 D10 A3	PE4 (4) PJ2 (10) PF4 (4) PB3 (2) PE1 (3) PH3 (2) PD6 (1)	I	TTL	PWM Fault 0.
Fault1	A7	PB6 (4)	I	TTL	PWM Fault 1.
Fault2	M1 F10	PC5 (4) PH5 (10)	I	TTL	PWM Fault 2.
Fault3	E11 D11	PB3 (4) PH2 (4)	I	TTL	PWM Fault 3.
PWM0	G1 F3 K1 L6 M9	PD0 (1) PJ0 (10) PG0 (2) PA6 (4) PF0 (3)	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
PWM1	G2 K2 M6 H12 B6	PD1 (1) PG1 (2) PA7 (4) PF1 (3) PJ1 (10)	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
PWM2	H2 J11 E12 C9	PD2 (3) PF2 (4) PB0 (2) PH0 (2)	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
PWM3	H1 J12 D12 C8	PD3 (3) PF3 (4) PB1 (2) PH1 (2)	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
PWM4	A1 K1 M4 L6 J11 G3 B11 C9	PE6 (1) PG0 (4) PA2 (4) PA6 (5) PF2 (2) PH6 (10) PE0 (1) PH0 (9)	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.

Table 21-2. Signals for PWM (108BGA) (continued)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
PWM5	B1 H3 K2 L4 M6 J12 A12 C8	PE7 (1) PH7 (10) PG1 (4) PA3 (4) PA7 (5) PF3 (2) PE1 (1) PH1 (9)	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.

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

21.3 Functional Description

21.3.1 PWM Timer

The timer in each PWM generator runs in one of two modes: Count-Down mode or Count-Up/Down mode. In Count-Down mode, the timer counts from the load value to zero, goes back to the load value, and continues counting down. In Count-Up/Down mode, the timer counts from zero up to the load value, back down to zero, back up to the load value, and so on. Generally, Count-Down mode is used for generating left- or right-aligned PWM signals, while the Count-Up/Down mode is used for generating center-aligned PWM signals.

The timers output three signals that are used in the PWM generation process: the direction signal (this is always Low in Count-Down mode, but alternates between Low and High in Count-Up/Down mode), a single-clock-cycle-width High pulse when the counter is zero, and a single-clock-cycle-width High pulse when the counter is equal to the load value. Note that in Count-Down mode, the zero pulse is immediately followed by the load pulse. In the figures in this chapter, these signals are labelled "dir," "zero," and "load."

21.3.2 PWM Comparators

Each PWM generator has two comparators that monitor the value of the counter; when either comparator matches the counter, they output a single-clock-cycle-width High pulse, labelled "cmpA" and "cmpB" in the figures in this chapter. When in Count-Up/Down mode, these comparators match both when counting up and when counting down, and thus are 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 1071 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 1071 shows the behavior of the counter and the relationship of these pulses when the counter is in Count-Up/Down mode. In these figures, the following definitions apply:

- LOAD is the value in the **PWMnLOAD** register
- COMPA is the value in the **PWMnCMPA** register
- COMPB is the value in the **PWMnCMPB** register
- 0 is the value zero
- load is the internal signal that has a single-clock-cycle-width High pulse when the counter is equal to the load value

- zero is the internal signal that has a single-clock-cycle-width High pulse when the counter is zero
- cmpA is the internal signal that has a single-clock-cycle-width High pulse when the counter is equal to COMPA
- cmpB is the internal signal that has a single-clock-cycle-width High pulse when the counter is equal to COMPB
- dir is the internal signal that indicates the count direction

Figure 21-3. PWM Count-Down Mode

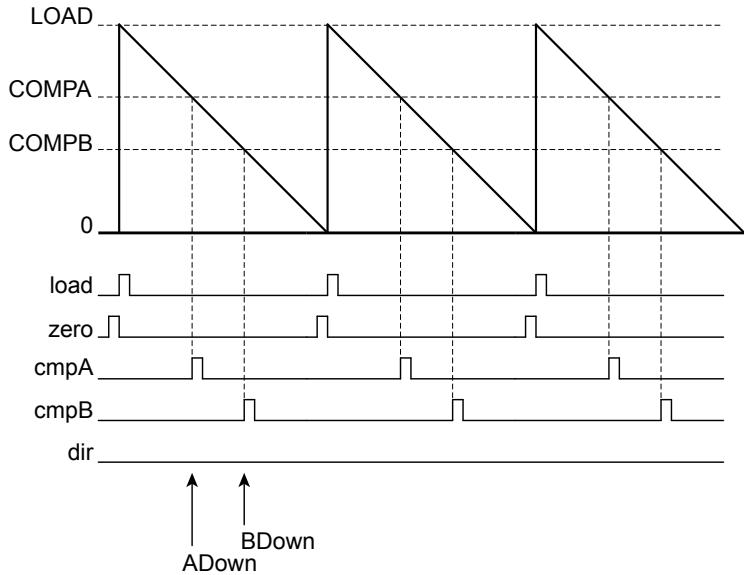
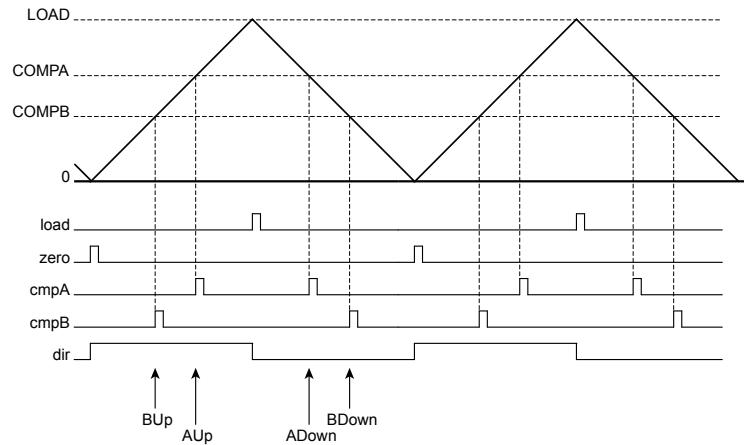


Figure 21-4. PWM Count-Up/Down Mode



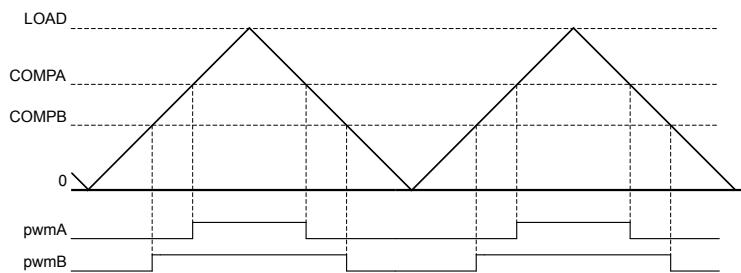
21.3.3 PWM Signal Generator

The PWM generator takes the load, zero, cmpA, and cmpB pulses (qualified by the dir signal) and generates two internal PWM signals, pwmA and pwmB. In Count-Down mode, there are four events that can affect these signals: zero, load, match A down, and match B down. In Count-Up/Down mode, there are six events that can affect these signals: 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 1072 shows the use of Count-Up/Down mode to generate a pair of center-aligned, overlapped PWM signals that have different duty cycles. This figure shows the pwmA and pwmB signals before they have passed through the dead-band generator.

Figure 21-5. PWM Generation Example In Count-Up/Down Mode



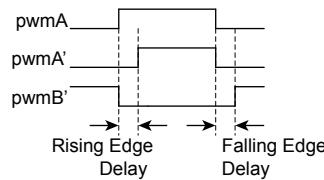
In this example, the first generator is set to drive High on match A up, drive Low on match A down, and ignore the other four events. The second generator is set to drive High on match B up, drive Low on match B down, and ignore the other four events. Changing the value of comparator A changes the duty cycle of the pwmA signal, and changing the value of comparator B changes the duty cycle of the pwmB signal.

21.3.4 Dead-Band Generator

The pwmA and pwmB signals produced by the PWM generator are passed to the dead-band generator. If the dead-band generator is disabled, the PWM signals simply pass through to the pwmA' and pwmB' signals unmodified. If the dead-band generator is enabled, the pwmB signal is lost and two PWM signals are generated based on the pwmA signal. The first output PWM signal, pwmA' is the pwmA signal with the rising edge delayed by a programmable amount. The second output PWM signal, pwmB', is the inversion of the pwmA signal with a programmable delay added between the falling edge of the pwmA signal and the rising edge of the pwmB' signal.

The resulting signals are 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 1072 shows the effect of the dead-band generator on the pwmA signal and the resulting pwmA' and pwmB' signals that are transmitted to the output control block.

Figure 21-6. PWM Dead-Band Generator



21.3.5 Interrupt/ADC-Trigger Selector

The PWM generator also takes the same four (or six) counter events and uses them to generate an interrupt or an ADC trigger. Any of these events or a set of these events can be selected as a source for an interrupt; when any of the selected events occur, an interrupt is generated. Additionally, the same event, a different event, the same set of events, or a different set of events can be selected as a source for an ADC trigger; when any of these selected events occur, an ADC trigger pulse is generated. The selection of events allows the interrupt or ADC trigger to occur at a specific position within the pwmA or pwmB 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.

21.3.6 Synchronization Methods

The PWM unit provides three PWM generators providing six PWM outputs that may be used in a wide variety of applications. Generally speaking, the PWM is used in one of two categories of operation:

- **Unsynchronized.** The PWM generator and its two output signals are used alone, 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, synchronization can be used to guarantee that they also have the same count value (the PWM generators must be configured before they are synchronized). With this feature, more than two PWM_n signals can be produced with a known relationship between the edges of those signals because 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 `SYNCn` bit associated with the generator. Multiple PWM generators can be synchronized together by setting all necessary `SYNCn` 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.

Additional synchronization can occur between multiple PWM generators by updating register contents 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 at the end of the PWM cycle. In this case, the effect of the write is deferred, providing a guaranteed defined behavior and preventing overly short or overly long output PWM pulses.
- **Globally Synchronized.** The write value does not affect the logic until two sequential events have occurred: (1) the Update mode for the generator function is programmed for global synchronization in the **PWM_nCTL** register, and (2) the counter reaches zero at the end of the PWM cycle. In this case, the effect of the write is deferred until the end of the PWM cycle 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 various Update mode bits and fields in the **PWMnCTL** register (LOADUPD; CMPAUPD; CMPBUPD):

- Generator Registers: **PWMnLOAD**, **PWMnCMPA**, and **PWMnCMPB**

The following registers default to immediate update, but are provided with the optional functionality of synchronously updating rather than having all updates take immediate effect:

- Module-Level Register: **PWMENABLE** (based on the state of the ENUPD_n bits in the PWMENUPD register).
- Generator Register: **PWMnGENA**, **PWMnGENB**, **PWMnDBCTL**, **PWMnDBRISE**, and **PWMnDBFALL** (based on the state of various Update mode bits and fields in the **PWMnCTL** register (GENAUPD; GENBUPD; DBCTLUPD; DBRISEUPD; DBFALLUPD)).

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.

21.3.7 Fault Conditions

A fault condition is one in which the controller must be signaled to stop normal PWM function and then set the PWM_n signals to a safe state. Two basic situations cause fault conditions:

- The microcontroller is stalled and cannot perform the necessary computation in the time required for motion control
- An external error or event is detected

The PWM unit can use the following inputs to generate a fault condition, including:

- FAULT_n 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.

Four fault input pins (FAULT0-FAULT3). These inputs may be used with circuits that generate an active High or active Low signal to indicate an error condition. A FAULT_n pins may be individually programmed for the appropriate logic sense using the **PWMnFLTSEN** register.

The PWM generator's mode control, including fault condition handling, is provided in the **PWMnCTL** register. This register determines whether the FAULT0 input or a combination of FAULT_n input signals and/or digital comparator triggers (as configured by the **PWMnFLTSRC0** and **PWMnFLTSRC1** registers) is used to generate a fault condition. The **PWMnCTL** register also selects whether the fault condition is maintained as long as the external condition lasts or if it is latched until the fault condition 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.

Status regarding the specific fault cause is provided in the **PWMnFLTSTAT0** and **PWMnFLTSTAT1** registers.

PWM generator fault conditions may be promoted to a controller interrupt using the **PWMINTEN** register.

21.3.8 Output Control Block

The output control block takes care of the final conditioning of the `pwmA'` and `pwmB'` signals before they go to the pins as the `PWMn` signals. Via a single register, the **PWM Output Enable (PWNENABLE)** register, the set of PWM signals that are actually enabled to the pins can be modified. This function 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). In addition, the updating of the bits in the **PWMENABLE** register can be configured to be immediate or locally or globally synchronized to the next synchronous update using the **PWM Enable Update (PWMENUPD)** register.

During fault conditions, the PWM output signals, `PWMn`, usually must be driven to safe values so that external equipment may be safely controlled. The **PWMFAULT** register specifies whether during a fault condition, the generated signal continues to be passed driven or to an encoding specified in the **PWMFAULTVAL** register.

A final inversion can be applied to any of the `PWMn` signals, making them active Low instead of the default active High using the **PWM Output Inversion (PWMINVERT)**. The inversion is applied even if a value has been enabled in the **PWMFAULT** register and specified in the **PWMFAULTVAL** register. In other words, if a bit is set in the **PWMFAULT**, **PWMFAULTVAL**, and **PWMINVERT** registers, the output on the `PWMn` signal is 0, not 1 as specified in the **PWMFAULTVAL** register.

21.4 Initialization and Configuration

The following example shows how to initialize PWM Generator 0 with a 25-kHz frequency, 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 268).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module (see page 285).
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register. To determine which GPIOs to configure, see Table 24-4 on page 1182.
4. Configure the `PMCn` fields in the **GPIOPCTL** register to assign the PWM signals to the appropriate pins (see page 465 and Table 24-5 on page 1190).
5. 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).
6. 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.
7. 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. Thus there are 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.
- 8. Set the pulse width of the **PWM0** pin for a 25% duty cycle.
 - Write the **PWM0CMPA** register with a value of 0x0000.012B.
- 9. Set the pulse width of the **PWM1** pin for a 75% duty cycle.
 - Write the **PWM0CMPB** register with a value of 0x0000.0063.
- 10. Start the timers in PWM generator 0.
 - Write the **PWM0CTL** register with a value of 0x0000.0001.
- 11. Enable PWM outputs.
 - Write the **PWMENABLE** register with a value of 0x0000.0003.

21.5 Register Map

Table 21-3 on page 1076 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 268).

Table 21-3. PWM Register Map

Offset	Name	Type	Reset	Description	See page
0x000	PWMCTL	R/W	0x0000.0000	PWM Master Control	1079
0x004	PWMSYNC	R/W	0x0000.0000	PWM Time Base Sync	1080
0x008	PWMENABLE	R/W	0x0000.0000	PWM Output Enable	1081
0x00C	PWMINVERT	R/W	0x0000.0000	PWM Output Inversion	1083
0x010	PWMFAULT	R/W	0x0000.0000	PWM Output Fault	1085
0x014	PWMINTEN	R/W	0x0000.0000	PWM Interrupt Enable	1087
0x018	PWMRIS	RO	0x0000.0000	PWM Raw Interrupt Status	1089
0x01C	PWMISC	R/W1C	0x0000.0000	PWM Interrupt Status and Clear	1091
0x020	PWMSTATUS	RO	0x0000.0000	PWM Status	1093
0x024	PWMFAULTVAL	R/W	0x0000.0000	PWM Fault Condition Value	1095
0x028	PWMENUPD	R/W	0x0000.0000	PWM Enable Update	1097
0x040	PWM0CTL	R/W	0x0000.0000	PWM0 Control	1100
0x044	PWM0INTEN	R/W	0x0000.0000	PWM0 Interrupt and Trigger Enable	1105
0x048	PWM0RIS	RO	0x0000.0000	PWM0 Raw Interrupt Status	1108
0x04C	PWM0ISC	R/W1C	0x0000.0000	PWM0 Interrupt Status and Clear	1110

Table 21-3. PWM Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x050	PWM0LOAD	R/W	0x0000.0000	PWM0 Load	1112
0x054	PWM0COUNT	RO	0x0000.0000	PWM0 Counter	1113
0x058	PWM0CMPA	R/W	0x0000.0000	PWM0 Compare A	1114
0x05C	PWM0CMPB	R/W	0x0000.0000	PWM0 Compare B	1115
0x060	PWM0GENA	R/W	0x0000.0000	PWM0 Generator A Control	1116
0x064	PWM0GENB	R/W	0x0000.0000	PWM0 Generator B Control	1119
0x068	PWM0DBCTL	R/W	0x0000.0000	PWM0 Dead-Band Control	1122
0x06C	PWM0DBRISE	R/W	0x0000.0000	PWM0 Dead-Band Rising-Edge Delay	1123
0x070	PWM0DBFALL	R/W	0x0000.0000	PWM0 Dead-Band Falling-Edge-Delay	1124
0x074	PWM0FLTSRC0	R/W	0x0000.0000	PWM0 Fault Source 0	1125
0x078	PWM0FLTSRC1	R/W	0x0000.0000	PWM0 Fault Source 1	1127
0x07C	PWM0MINFLTPER	R/W	0x0000.0000	PWM0 Minimum Fault Period	1130
0x080	PWM1CTL	R/W	0x0000.0000	PWM1 Control	1100
0x084	PWM1INTEN	R/W	0x0000.0000	PWM1 Interrupt and Trigger Enable	1105
0x088	PWM1RIS	RO	0x0000.0000	PWM1 Raw Interrupt Status	1108
0x08C	PWM1ISC	R/W1C	0x0000.0000	PWM1 Interrupt Status and Clear	1110
0x090	PWM1LOAD	R/W	0x0000.0000	PWM1 Load	1112
0x094	PWM1COUNT	RO	0x0000.0000	PWM1 Counter	1113
0x098	PWM1CMPA	R/W	0x0000.0000	PWM1 Compare A	1114
0x09C	PWM1CMPB	R/W	0x0000.0000	PWM1 Compare B	1115
0x0A0	PWM1GENA	R/W	0x0000.0000	PWM1 Generator A Control	1116
0x0A4	PWM1GENB	R/W	0x0000.0000	PWM1 Generator B Control	1119
0x0A8	PWM1DBCTL	R/W	0x0000.0000	PWM1 Dead-Band Control	1122
0x0AC	PWM1DBRISE	R/W	0x0000.0000	PWM1 Dead-Band Rising-Edge Delay	1123
0x0B0	PWM1DBFALL	R/W	0x0000.0000	PWM1 Dead-Band Falling-Edge-Delay	1124
0x0B4	PWM1FLTSRC0	R/W	0x0000.0000	PWM1 Fault Source 0	1125
0x0B8	PWM1FLTSRC1	R/W	0x0000.0000	PWM1 Fault Source 1	1127
0x0BC	PWM1MINFLTPER	R/W	0x0000.0000	PWM1 Minimum Fault Period	1130
0x0C0	PWM2CTL	R/W	0x0000.0000	PWM2 Control	1100
0x0C4	PWM2INTEN	R/W	0x0000.0000	PWM2 Interrupt and Trigger Enable	1105
0x0C8	PWM2RIS	RO	0x0000.0000	PWM2 Raw Interrupt Status	1108
0x0CC	PWM2ISC	R/W1C	0x0000.0000	PWM2 Interrupt Status and Clear	1110

Table 21-3. PWM Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x0D0	PWM2LOAD	R/W	0x0000.0000	PWM2 Load	1112
0x0D4	PWM2COUNT	RO	0x0000.0000	PWM2 Counter	1113
0x0D8	PWM2CMPA	R/W	0x0000.0000	PWM2 Compare A	1114
0x0DC	PWM2CMPB	R/W	0x0000.0000	PWM2 Compare B	1115
0x0E0	PWM2GENA	R/W	0x0000.0000	PWM2 Generator A Control	1116
0x0E4	PWM2GENB	R/W	0x0000.0000	PWM2 Generator B Control	1119
0x0E8	PWM2DBCTL	R/W	0x0000.0000	PWM2 Dead-Band Control	1122
0x0EC	PWM2DBRISE	R/W	0x0000.0000	PWM2 Dead-Band Rising-Edge Delay	1123
0x0F0	PWM2DBFALL	R/W	0x0000.0000	PWM2 Dead-Band Falling-Edge-Delay	1124
0x0F4	PWM2FLTSRC0	R/W	0x0000.0000	PWM2 Fault Source 0	1125
0x0F8	PWM2FLTSRC1	R/W	0x0000.0000	PWM2 Fault Source 1	1127
0x0FC	PWM2MINFLTPER	R/W	0x0000.0000	PWM2 Minimum Fault Period	1130
0x800	PWM0FLTSEN	R/W	0x0000.0000	PWM0 Fault Pin Logic Sense	1131
0x804	PWM0FLTSTAT0	-	0x0000.0000	PWM0 Fault Status 0	1132
0x808	PWM0FLTSTAT1	-	0x0000.0000	PWM0 Fault Status 1	1134
0x880	PWM1FLTSEN	R/W	0x0000.0000	PWM1 Fault Pin Logic Sense	1131
0x884	PWM1FLTSTAT0	-	0x0000.0000	PWM1 Fault Status 0	1132
0x888	PWM1FLTSTAT1	-	0x0000.0000	PWM1 Fault Status 1	1134
0x900	PWM2FLTSEN	R/W	0x0000.0000	PWM2 Fault Pin Logic Sense	1131
0x904	PWM2FLTSTAT0	-	0x0000.0000	PWM2 Fault Status 0	1132
0x908	PWM2FLTSTAT1	-	0x0000.0000	PWM2 Fault Status 1	1134
0x980	PWM3FLTSEN	R/W	0x0000.0000	PWM3 Fault Pin Logic Sense	1131

21.6 Register Descriptions

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

Register 1: PWM Master Control (PWMCTL), offset 0x000

This register provides master control over the PWM generation blocks.

PWM Master Control (PWMCTL)

Base 0x4002.8000
Offset 0x000
Type R/W, reset 0x0000.0000

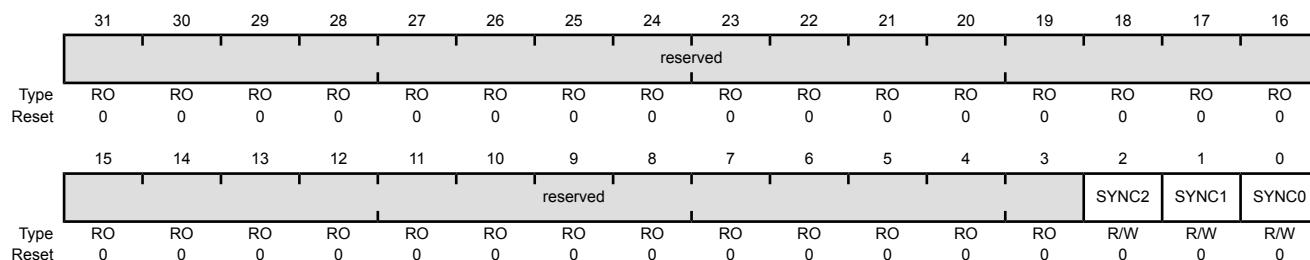
Bit/Field	Name	Type	Reset	Description						
31:3	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
2	GLOBALSYNC2	R/W	0	<p>Update PWM Generator 2</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Any queued update to a load or comparator register in PWM generator 2 is applied the next time the corresponding counter becomes zero.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table> <p>This bit automatically clears when the updates have completed; it cannot be cleared by software.</p>	Value	Description	1	Any queued update to a load or comparator register in PWM generator 2 is applied the next time the corresponding counter becomes zero.	0	No effect.
Value	Description									
1	Any queued update to a load or comparator register in PWM generator 2 is applied the next time the corresponding counter becomes zero.									
0	No effect.									
1	GLOBALSYNC1	R/W	0	<p>Update PWM Generator 1</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Any queued update to a load or comparator register in PWM generator 1 is applied the next time the corresponding counter becomes zero.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table> <p>This bit automatically clears when the updates have completed; it cannot be cleared by software.</p>	Value	Description	1	Any queued update to a load or comparator register in PWM generator 1 is applied the next time the corresponding counter becomes zero.	0	No effect.
Value	Description									
1	Any queued update to a load or comparator register in PWM generator 1 is applied the next time the corresponding counter becomes zero.									
0	No effect.									
0	GLOBALSYNC0	R/W	0	<p>Update PWM Generator 0</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>Any queued update to a load or comparator register in PWM generator 0 is applied the next time the corresponding counter becomes zero.</td></tr> <tr> <td>0</td><td>No effect.</td></tr> </tbody> </table> <p>This bit automatically clears when the updates have completed; it cannot be cleared by software.</p>	Value	Description	1	Any queued update to a load or comparator register in PWM generator 0 is applied the next time the corresponding counter becomes zero.	0	No effect.
Value	Description									
1	Any queued update to a load or comparator register in PWM generator 0 is applied the next time the corresponding counter becomes zero.									
0	No effect.									

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. Setting a bit in this register causes the specified counter to reset back to 0; setting multiple bits resets multiple counters simultaneously. The bits auto-clear after the reset has occurred; reading them back as zero indicates that the synchronization has completed.

PWM Time Base Sync (PWMSYNC)

Base 0x4002.8000
Offset 0x004
Type R/W, reset 0x0000.0000



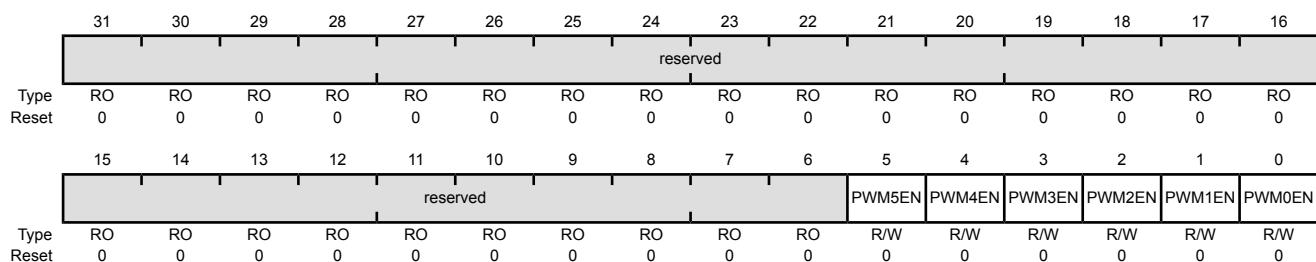
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	SYNC2	R/W	0	Reset Generator 2 Counter
		Value	Description	
		1	Resets the PWM generator 2 counter.	
		0	No effect.	
1	SYNC1	R/W	0	Reset Generator 1 Counter
		Value	Description	
		1	Resets the PWM generator 1 counter.	
		0	No effect.	
0	SYNC0	R/W	0	Reset Generator 0 Counter
		Value	Description	
		1	Resets the PWM generator 0 counter.	
		0	No effect.	

Register 3: PWM Output Enable (PWMENABLE), offset 0x008

This register provides a master control of which generated pwmA' and pwmB' signals are output to the `PWMn` 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 pwmA' or pwmB' signal is passed through to the output stage. When bits are clear, the pwmA' or pwmB' signal is replaced by a zero value which is also passed to the output stage. The **PWMINVERT** register controls the output stage, so if the corresponding bit is set in that register, the value seen on the `PWMn` signal is inverted from what is configured by the bits in this register. Updates to the bits in this register can be immediate or locally or globally synchronized to the next synchronous update as controlled by the `ENUPDn` fields in the **PWMENUPD** register.

PWM Output Enable (PWMENABLE)

Base 0x4002.8000
Offset 0x008
Type R/W, reset 0x0000.0000



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	PWM5EN	R/W	0	PWM5 Output Enable
				Value Description
				1 The generated pwm2B' signal is passed to the <code>PWM5</code> pin.
				0 The <code>PWM5</code> signal has a zero value.
4	PWM4EN	R/W	0	PWM4 Output Enable
				Value Description
				1 The generated pwm2A' signal is passed to the <code>PWM4</code> pin.
				0 The <code>PWM4</code> signal has a zero value.
3	PWM3EN	R/W	0	PWM3 Output Enable
				Value Description
				1 The generated pwm1B' signal is passed to the <code>PWM3</code> pin.
				0 The <code>PWM3</code> signal has a zero value.

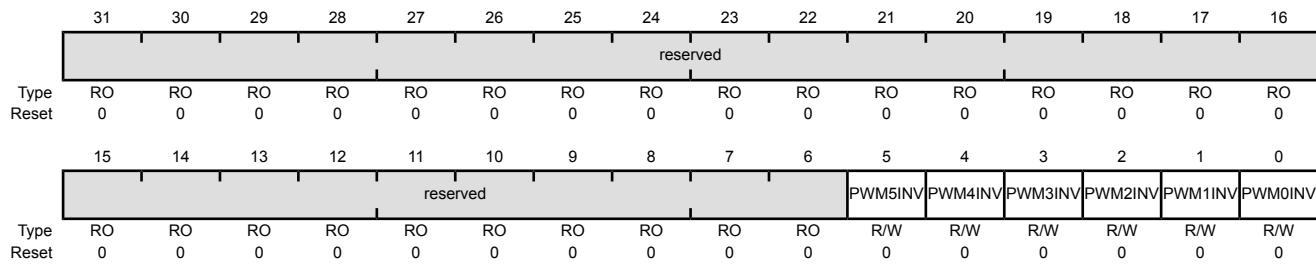
Bit/Field	Name	Type	Reset	Description
2	PWM2EN	R/W	0	PWM2 Output Enable Value Description 1 The generated pwm1A' signal is passed to the PWM2 pin. 0 The PWM2 signal has a zero value.
1	PWM1EN	R/W	0	PWM1 Output Enable Value Description 1 The generated pwm0B' signal is passed to the PWM1 pin. 0 The PWM1 signal has a zero value.
0	PWM0EN	R/W	0	PWM0 Output Enable Value Description 1 The generated pwm0A' signal is passed to the PWM0 pin. 0 The PWM0 signal has a zero value.

Register 4: PWM Output Inversion (P威MINVERT), offset 0x00C

This register provides a master control of the polarity of the `PWMn` signals on the device pins. The `pwmA'` and `pwmB'` signals generated by the PWM generator are active High; but can be made active Low via this register. Disabled PWM channels are also passed through the output inverter (if so configured) so that inactive signals can be High. In addition, if the **PWMFAULT** register enables a specific value to be placed on the `PWMn` signals during a fault condition, that value is inverted if the corresponding bit in this register is set.

PWM Output Inversion (P威MINVERT)

Base 0x4002.8000
Offset 0x00C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	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	PWM5INV	R/W	0	Invert PWM5 Signal
				Value Description
			1	The PWM5 signal is inverted.
			0	The PWM5 signal is not inverted.
4	PWM4INV	R/W	0	Invert PWM4 Signal
				Value Description
			1	The PWM4 signal is inverted.
			0	The PWM4 signal is not inverted.
3	PWM3INV	R/W	0	Invert PWM3 Signal
				Value Description
			1	The PWM3 signal is inverted.
			0	The PWM3 signal is not inverted.
2	PWM2INV	R/W	0	Invert PWM2 Signal
				Value Description
			1	The PWM2 signal is inverted.
			0	The PWM2 signal is not inverted.

Bit/Field	Name	Type	Reset	Description
1	PWM1INV	R/W	0	Invert PWM1 Signal
				Value Description
			1	The PWM1 signal is inverted.
			0	The PWM1 signal is not inverted.
0	PWM0INV	R/W	0	Invert PWM0 Signal
				Value Description
			1	The PWM0 signal is inverted.
			0	The PWM0 signal is not inverted.

Register 5: PWM Output Fault (PWMFAULT), offset 0x010

This register controls the behavior of the `PWMn` outputs in the presence of fault conditions. Both the fault inputs (FAULTn pins and digital comparator outputs) and debug events are considered fault conditions. On a fault condition, each `pwmA'` or `pwmB'` signal can be passed through unmodified or driven to the value specified by the corresponding bit in the **PWMFAULTVAL** register. For outputs that are configured for pass-through, the debug event handling on the corresponding PWM generator also determines if the `pwmA'` or `pwmB'` 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

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

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	FAULT5	R/W	0	PWM5 Fault
		Value	Description	
		1	The PWM5 output signal is driven to the value specified by the PWM5 bit in the PWMFAULTVAL register.	
		0	The generated pwm2B' signal is passed to the PWM5 pin.	
4	FAULT4	R/W	0	PWM4 Fault
		Value	Description	
		1	The PWM4 output signal is driven to the value specified by the PWM4 bit in the PWMFAULTVAL register.	
		0	The generated pwm2A' signal is passed to the PWM4 pin.	
3	FAULT3	R/W	0	PWM3 Fault
		Value	Description	
		1	The PWM3 output signal is driven to the value specified by the PWM3 bit in the PWMFAULTVAL register.	
		0	The generated pwm1B' signal is passed to the PWM3 pin.	

Bit/Field	Name	Type	Reset	Description
2	FAULT2	R/W	0	PWM2 Fault Value Description 1 The PWM2 output signal is driven to the value specified by the PWM2 bit in the PWMFAULTVAL register. 0 The generated pwm1A' signal is passed to the PWM2 pin.
1	FAULT1	R/W	0	PWM1 Fault Value Description 1 The PWM1 output signal is driven to the value specified by the PWM1 bit in the PWMFAULTVAL register. 0 The generated pwm0B' signal is passed to the PWM1 pin.
0	FAULT0	R/W	0	PWM0 Fault Value Description 1 The PWM0 output signal is driven to the value specified by the PWM0 bit in the PWMFAULTVAL register. 0 The generated pwm0A' signal is passed to the PWM0 pin.

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
Type	RO	R/W	R/W	R/W	R/W											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO	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	INTFAULT3	R/W	0	Interrupt Fault 3
		Value	Description	
		1	An interrupt is sent to the interrupt controller when the fault condition for PWM generator 3 is asserted.	
		0	The fault condition for PWM generator 3 is suppressed and not sent to the interrupt controller.	
18	INTFAULT2	R/W	0	Interrupt Fault 2
		Value	Description	
		1	An interrupt is sent to the interrupt controller when the fault condition for PWM generator 2 is asserted.	
		0	The fault condition for PWM generator 2 is suppressed and not sent to the interrupt controller.	
17	INTFAULT1	R/W	0	Interrupt Fault 1
		Value	Description	
		1	An interrupt is sent to the interrupt controller when the fault condition for PWM generator 1 is asserted.	
		0	The fault condition for PWM generator 1 is suppressed and not sent to the interrupt controller.	

Bit/Field	Name	Type	Reset	Description
16	INTFAULT0	R/W	0	Interrupt Fault 0 Value Description 1 An interrupt is sent to the interrupt controller when the fault condition for PWM generator 0 is asserted. 0 The fault condition for PWM generator 0 is suppressed and not sent to the interrupt controller.
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	INTPWM2	R/W	0	PWM2 Interrupt Enable Value Description 1 An interrupt is sent to the interrupt controller when the PWM generator 2 block asserts an interrupt. 0 The PWM generator 2 interrupt is suppressed and not sent to the interrupt controller.
1	INTPWM1	R/W	0	PWM1 Interrupt Enable Value Description 1 An interrupt is sent to the interrupt controller when the PWM generator 1 block asserts an interrupt. 0 The PWM generator 1 interrupt is suppressed and not sent to the interrupt controller.
0	INTPWM0	R/W	0	PWM0 Interrupt Enable Value Description 1 An interrupt is sent to the interrupt controller when the PWM generator 0 block asserts an interrupt. 0 The PWM generator 0 interrupt is suppressed and not sent to the interrupt controller.

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 are enabled to cause an interrupt to be asserted to the interrupt controller. The fault interrupt is asserted based on the fault condition source that is specified by the **PWMnCTL**, **PWMnFLTSRC0** and **PWMnFLTSRC1** registers. The fault interrupt is latched on detection and must be cleared through the **PWM Interrupt Status and Clear (PWMISC)** register. The actual value of the FAULTn signals can be observed using the **PWMSTATUS** register.

The PWM generator interrupts simply reflect the status of the PWM generators and are cleared via the interrupt status register in the PWM generator blocks. If a bit is set, the event is active; if a bit is clear the event 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
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved																
Type	RO															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description						
31:20	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	INTFAULT3	RO	0	<p>Interrupt Fault PWM 3</p> <table border="0"> <tr> <td style="vertical-align: top; padding-right: 10px;">Value</td> <td>Description</td> </tr> <tr> <td>1</td> <td>The fault condition for PWM generator 3 is asserted.</td> </tr> <tr> <td>0</td> <td>The fault condition for PWM generator 3 has not been asserted.</td> </tr> </table> <p>This bit is cleared by writing a 1 to the INTFAULT3 bit in the PWMISC register.</p>	Value	Description	1	The fault condition for PWM generator 3 is asserted.	0	The fault condition for PWM generator 3 has not been asserted.
Value	Description									
1	The fault condition for PWM generator 3 is asserted.									
0	The fault condition for PWM generator 3 has not been asserted.									
18	INTFAULT2	RO	0	<p>Interrupt Fault PWM 2</p> <table border="0"> <tr> <td style="vertical-align: top; padding-right: 10px;">Value</td> <td>Description</td> </tr> <tr> <td>1</td> <td>The fault condition for PWM generator 2 is asserted.</td> </tr> <tr> <td>0</td> <td>The fault condition for PWM generator 2 has not been asserted.</td> </tr> </table> <p>This bit is cleared by writing a 1 to the INTFAULT2 bit in the PWMISC register.</p>	Value	Description	1	The fault condition for PWM generator 2 is asserted.	0	The fault condition for PWM generator 2 has not been asserted.
Value	Description									
1	The fault condition for PWM generator 2 is asserted.									
0	The fault condition for PWM generator 2 has not been asserted.									

Bit/Field	Name	Type	Reset	Description
17	INTFAULT1	RO	0	<p>Interrupt Fault PWM 1</p> <p>Value Description</p> <p>1 The fault condition for PWM generator 1 is asserted.</p> <p>0 The fault condition for PWM generator 1 has not been asserted.</p> <p>This bit is cleared by writing a 1 to the INTFAULT1 bit in the PWMISC register.</p>
16	INTFAULT0	RO	0	<p>Interrupt Fault PWM 0</p> <p>Value Description</p> <p>1 The fault condition for PWM generator 0 is asserted.</p> <p>0 The fault condition for PWM generator 0 has not been asserted.</p> <p>This bit is cleared by writing a 1 to the INTFAULT0 bit in the PWMISC register.</p>
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	INTPWM2	RO	0	<p>PWM2 Interrupt Asserted</p> <p>Value Description</p> <p>1 The PWM generator 2 block interrupt is asserted.</p> <p>0 The PWM generator 2 block interrupt has not been asserted.</p> <p>The PWM2RIS register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM2ISC register.</p>
1	INTPWM1	RO	0	<p>PWM1 Interrupt Asserted</p> <p>Value Description</p> <p>1 The PWM generator 1 block interrupt is asserted.</p> <p>0 The PWM generator 1 block interrupt has not been asserted.</p> <p>The PWM1RIS register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM1ISC register.</p>
0	INTPWM0	RO	0	<p>PWM0 Interrupt Asserted</p> <p>Value Description</p> <p>1 The PWM generator 0 block interrupt is asserted.</p> <p>0 The PWM generator 0 block interrupt has not been asserted.</p> <p>The PWM0RIS register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM0ISC register.</p>

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. If a fault interrupt is set, the corresponding FAULTn input has caused an interrupt. For the fault interrupt, a write of 1 to that bit position clears the latched interrupt status. If an block interrupt bit is set, the corresponding generator block is asserting an interrupt. The individual interrupt status registers, **PWMnISC**, in each block must be consulted to determine the reason for the interrupt and used to clear the interrupt.

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												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	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	INTFAULT3	R/W1C	0	<p>FAULT3 Interrupt Asserted</p> <p>Value Description</p> <p>1 An enabled interrupt for the fault condition for PWM generator 3 is asserted or is latched.</p> <p>0 The fault condition for PWM generator 3 has not been asserted or is not enabled.</p> <p>Writing a 1 to this bit clears it and the INTFAULT3 bit in the PWMRIS register.</p>
18	INTFAULT2	R/W1C	0	<p>FAULT2 Interrupt Asserted</p> <p>Value Description</p> <p>1 An enabled interrupt for the fault condition for PWM generator 2 is asserted or is latched.</p> <p>0 The fault condition for PWM generator 2 has not been asserted or is not enabled.</p> <p>Writing a 1 to this bit clears it and the INTFAULT2 bit in the PWMRIS register.</p>

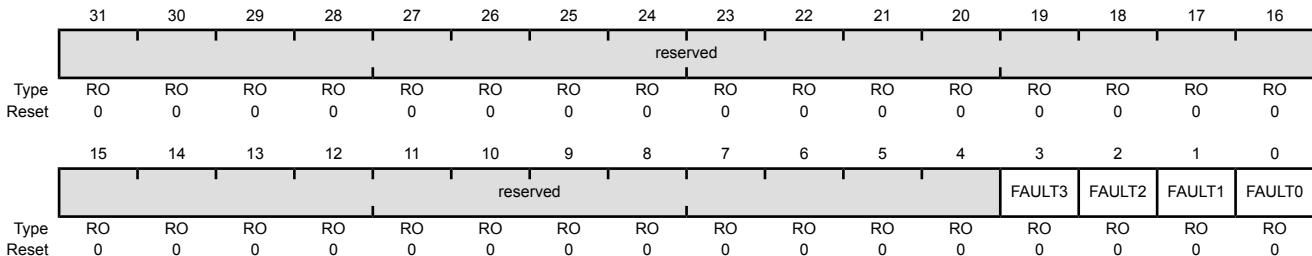
Bit/Field	Name	Type	Reset	Description						
17	INTFAULT1	R/W1C	0	<p>FAULT1 Interrupt Asserted</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An enabled interrupt for the fault condition for PWM generator 1 is asserted or is latched.</td></tr> <tr> <td>0</td><td>The fault condition for PWM generator 1 has not been asserted or is not enabled.</td></tr> </tbody> </table> <p>Writing a 1 to this bit clears it and the INTFAULT1 bit in the PWMRIS register.</p>	Value	Description	1	An enabled interrupt for the fault condition for PWM generator 1 is asserted or is latched.	0	The fault condition for PWM generator 1 has not been asserted or is not enabled.
Value	Description									
1	An enabled interrupt for the fault condition for PWM generator 1 is asserted or is latched.									
0	The fault condition for PWM generator 1 has not been asserted or is not enabled.									
16	INTFAULT0	R/W1C	0	<p>FAULT0 Interrupt Asserted</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An enabled interrupt for the fault condition for PWM generator 0 is asserted or is latched.</td></tr> <tr> <td>0</td><td>The fault condition for PWM generator 0 has not been asserted or is not enabled.</td></tr> </tbody> </table> <p>Writing a 1 to this bit clears it and the INTFAULT0 bit in the PWMRIS register.</p>	Value	Description	1	An enabled interrupt for the fault condition for PWM generator 0 is asserted or is latched.	0	The fault condition for PWM generator 0 has not been asserted or is not enabled.
Value	Description									
1	An enabled interrupt for the fault condition for PWM generator 0 is asserted or is latched.									
0	The fault condition for PWM generator 0 has not been asserted or is not enabled.									
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	INTPWM2	RO	0	<p>PWM2 Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An enabled interrupt for the PWM generator 2 block is asserted.</td></tr> <tr> <td>0</td><td>The PWM generator 2 block interrupt is not asserted or is not enabled.</td></tr> </tbody> </table> <p>The PWM2RIS register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM2ISC register.</p>	Value	Description	1	An enabled interrupt for the PWM generator 2 block is asserted.	0	The PWM generator 2 block interrupt is not asserted or is not enabled.
Value	Description									
1	An enabled interrupt for the PWM generator 2 block is asserted.									
0	The PWM generator 2 block interrupt is not asserted or is not enabled.									
1	INTPWM1	RO	0	<p>PWM1 Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An enabled interrupt for the PWM generator 1 block is asserted.</td></tr> <tr> <td>0</td><td>The PWM generator 1 block interrupt is not asserted or is not enabled.</td></tr> </tbody> </table> <p>The PWM1RIS register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM1ISC register.</p>	Value	Description	1	An enabled interrupt for the PWM generator 1 block is asserted.	0	The PWM generator 1 block interrupt is not asserted or is not enabled.
Value	Description									
1	An enabled interrupt for the PWM generator 1 block is asserted.									
0	The PWM generator 1 block interrupt is not asserted or is not enabled.									
0	INTPWM0	RO	0	<p>PWM0 Interrupt Status</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>1</td><td>An enabled interrupt for the PWM generator 0 block is asserted.</td></tr> <tr> <td>0</td><td>The PWM generator 0 block interrupt is not asserted or is not enabled.</td></tr> </tbody> </table> <p>The PWM0RIS register shows the source of this interrupt. This bit is cleared by writing a 1 to the corresponding bit in the PWM0ISC register.</p>	Value	Description	1	An enabled interrupt for the PWM generator 0 block is asserted.	0	The PWM generator 0 block interrupt is not asserted or is not enabled.
Value	Description									
1	An enabled interrupt for the PWM generator 0 block is asserted.									
0	The PWM generator 0 block interrupt is not asserted or is not enabled.									

Register 9: PWM Status (PWMSTATUS), offset 0x020

This register provides the unlatched status of the PWM generator fault condition.

PWM Status (PWMSTATUS)

Base 0x4002.8000
Offset 0x020
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	FAULT3	RO	0	Generator 3 Fault Status
		Value	Description	
		1	The fault condition for PWM generator 3 is asserted. If the FLTSRC bit in the PWM3CTL register is clear, the FAULT0 input is the source of the fault condition, and is therefore asserted.	
		0	The fault condition for PWM generator 3 is not asserted.	
2	FAULT2	RO	0	Generator 2 Fault Status
		Value	Description	
		1	The fault condition for PWM generator 2 is asserted. If the FLTSRC bit in the PWM2CTL register is clear, the FAULT0 input is the source of the fault condition, and is therefore asserted.	
		0	The fault condition for PWM generator 2 is not asserted.	
1	FAULT1	RO	0	Generator 1 Fault Status
		Value	Description	
		1	The fault condition for PWM generator 1 is asserted. If the FLTSRC bit in the PWM1CTL register is clear, the FAULT0 input is the source of the fault condition, and is therefore asserted.	
		0	The fault condition for PWM generator 1 is not asserted.	

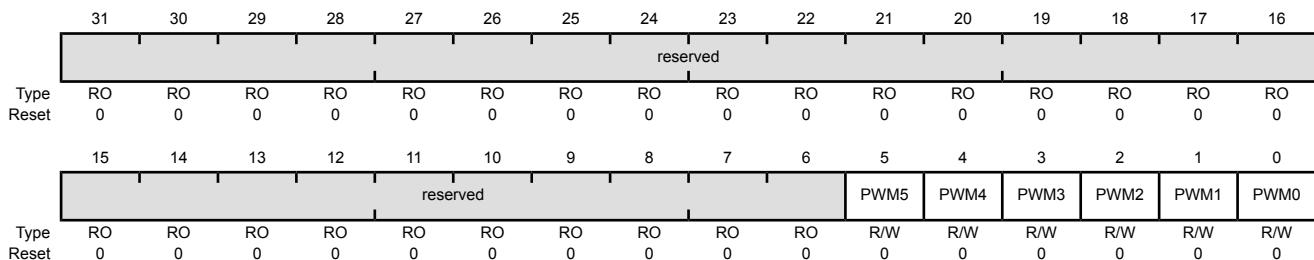
Bit/Field	Name	Type	Reset	Description
0	FAULT0	RO	0	Generator 0 Fault Status
Value Description				
			1	The fault condition for PWM generator 0 is asserted. If the FLTSRC bit in the PWM0CTL register is clear, the FAULT0 input is the source of the fault condition, and is therefore asserted.
			0	The fault condition for PWM generator 0 is not asserted.

Register 10: PWM Fault Condition Value (PWMFAULTVAL), offset 0x024

This register specifies the output value driven on the `PWMn` signals during a fault condition if enabled by the corresponding bit in the **PWMFAULT** register. Note that if the corresponding bit in the **PWMINVERT** register is set, the output value is driven to the logical NOT of the bit value in this register.

PWM Fault Condition Value (PWMFAULTVAL)

Base 0x4002.8000
Offset 0x024
Type R/W, reset 0x0000.0000



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	PWM5	R/W	0	PWM5 Fault Value
	Value	Description		
	1	The PWM5 output signal is driven High during fault conditions if the FAULT5 bit in the PWMFAULT register is set.		
	0	The PWM5 output signal is driven Low during fault conditions if the FAULT5 bit in the PWMFAULT register is set.		
4	PWM4	R/W	0	PWM4 Fault Value
	Value	Description		
	1	The PWM4 output signal is driven High during fault conditions if the FAULT4 bit in the PWMFAULT register is set.		
	0	The PWM4 output signal is driven Low during fault conditions if the FAULT4 bit in the PWMFAULT register is set.		
3	PWM3	R/W	0	PWM3 Fault Value
	Value	Description		
	1	The PWM3 output signal is driven High during fault conditions if the FAULT3 bit in the PWMFAULT register is set.		
	0	The PWM3 output signal is driven Low during fault conditions if the FAULT3 bit in the PWMFAULT register is set.		

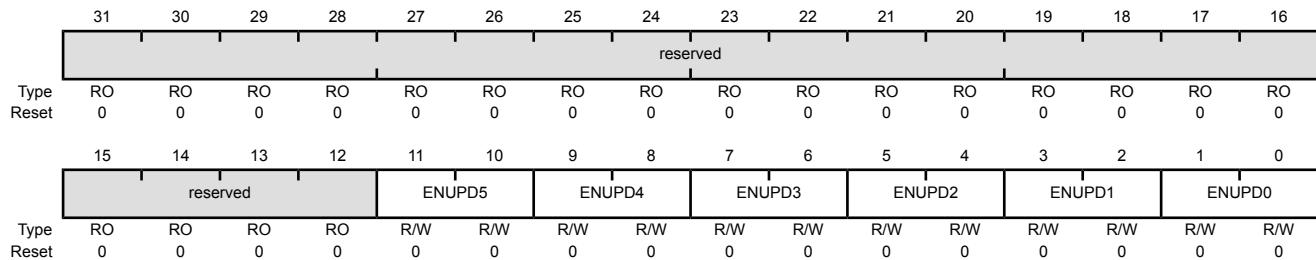
Bit/Field	Name	Type	Reset	Description
2	PWM2	R/W	0	PWM2 Fault Value Value Description 1 The PWM2 output signal is driven High during fault conditions if the FAULT2 bit in the PWMFAULT register is set. 0 The PWM2 output signal is driven Low during fault conditions if the FAULT2 bit in the PWMFAULT register is set.
1	PWM1	R/W	0	PWM1 Fault Value Value Description 1 The PWM1 output signal is driven High during fault conditions if the FAULT1 bit in the PWMFAULT register is set. 0 The PWM1 output signal is driven Low during fault conditions if the FAULT1 bit in the PWMFAULT register is set.
0	PWM0	R/W	0	PWM0 Fault Value Value Description 1 The PWM0 output signal is driven High during fault conditions if the FAULT0 bit in the PWMFAULT register is set. 0 The PWM0 output signal is driven Low during fault conditions if the FAULT0 bit in the PWMFAULT register is set.

Register 11: PWM Enable Update (PWMMENUPD), offset 0x028

This register specifies when updates to the `PWMnEn` bit in the **PWMENABLE** register are performed. The `PWMnEn` bit enables the `pwmA` or `pwmB` output to be passed to the microcontroller's pin. Updates can be immediate or locally or globally synchronized to the next synchronous update.

PWM Enable Update (PWMMENUPD)

Base 0x4002.8000
Offset 0x028
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11:10	ENUPD5	R/W	0	PWM5 Enable Update Mode
	Value	Description		
	0x0	Immediate		Writes to the <code>PWM5En</code> bit in the PWMENABLE register are used by the PWM generator module immediately.
	0x1	Reserved		
	0x2	Locally Synchronized		Writes to the <code>PWM5En</code> bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0.
	0x3	Globally Synchronized		Writes to the <code>PWM5En</code> bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.

Bit/Field	Name	Type	Reset	Description										
9:8	ENUPD4	R/W	0	PWM4 Enable Update Mode										
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Immediate Writes to the PWM4En bit in the PWMENABLE register are used by the PWM generator module immediately.</td></tr> <tr> <td>0x1</td><td>Reserved</td></tr> <tr> <td>0x2</td><td>Locally Synchronized Writes to the PWM4En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0.</td></tr> <tr> <td>0x3</td><td>Globally Synchronized Writes to the PWM4En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.</td></tr> </tbody> </table>	Value	Description	0x0	Immediate Writes to the PWM4En bit in the PWMENABLE register are used by the PWM generator module immediately.	0x1	Reserved	0x2	Locally Synchronized Writes to the PWM4En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0.	0x3	Globally Synchronized Writes to the PWM4En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.
Value	Description													
0x0	Immediate Writes to the PWM4En bit in the PWMENABLE register are used by the PWM generator module immediately.													
0x1	Reserved													
0x2	Locally Synchronized Writes to the PWM4En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0.													
0x3	Globally Synchronized Writes to the PWM4En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.													
7:6	ENUPD3	R/W	0	PWM3 Enable Update Mode										
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Immediate Writes to the PWM3En bit in the PWMENABLE register are used by the PWM generator module immediately.</td></tr> <tr> <td>0x1</td><td>Reserved</td></tr> <tr> <td>0x2</td><td>Locally Synchronized Writes to the PWM3En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0.</td></tr> <tr> <td>0x3</td><td>Globally Synchronized Writes to the PWM3En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.</td></tr> </tbody> </table>	Value	Description	0x0	Immediate Writes to the PWM3En bit in the PWMENABLE register are used by the PWM generator module immediately.	0x1	Reserved	0x2	Locally Synchronized Writes to the PWM3En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0.	0x3	Globally Synchronized Writes to the PWM3En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.
Value	Description													
0x0	Immediate Writes to the PWM3En bit in the PWMENABLE register are used by the PWM generator module immediately.													
0x1	Reserved													
0x2	Locally Synchronized Writes to the PWM3En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0.													
0x3	Globally Synchronized Writes to the PWM3En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.													
5:4	ENUPD2	R/W	0	PWM2 Enable Update Mode										
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Immediate Writes to the PWM2En bit in the PWMENABLE register are used by the PWM generator module immediately.</td></tr> <tr> <td>0x1</td><td>Reserved</td></tr> <tr> <td>0x2</td><td>Locally Synchronized Writes to the PWM2En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0.</td></tr> <tr> <td>0x3</td><td>Globally Synchronized Writes to the PWM2En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.</td></tr> </tbody> </table>	Value	Description	0x0	Immediate Writes to the PWM2En bit in the PWMENABLE register are used by the PWM generator module immediately.	0x1	Reserved	0x2	Locally Synchronized Writes to the PWM2En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0.	0x3	Globally Synchronized Writes to the PWM2En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.
Value	Description													
0x0	Immediate Writes to the PWM2En bit in the PWMENABLE register are used by the PWM generator module immediately.													
0x1	Reserved													
0x2	Locally Synchronized Writes to the PWM2En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0.													
0x3	Globally Synchronized Writes to the PWM2En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.													

Bit/Field	Name	Type	Reset	Description										
3:2	ENUPD1	R/W	0	PWM1 Enable Update Mode										
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Immediate Writes to the PWM1En bit in the PWMENABLE register are used by the PWM generator module immediately.</td></tr> <tr> <td>0x1</td><td>Reserved</td></tr> <tr> <td>0x2</td><td>Locally Synchronized Writes to the PWM1En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0.</td></tr> <tr> <td>0x3</td><td>Globally Synchronized Writes to the PWM1En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.</td></tr> </tbody> </table>	Value	Description	0x0	Immediate Writes to the PWM1En bit in the PWMENABLE register are used by the PWM generator module immediately.	0x1	Reserved	0x2	Locally Synchronized Writes to the PWM1En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0.	0x3	Globally Synchronized Writes to the PWM1En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.
Value	Description													
0x0	Immediate Writes to the PWM1En bit in the PWMENABLE register are used by the PWM generator module immediately.													
0x1	Reserved													
0x2	Locally Synchronized Writes to the PWM1En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0.													
0x3	Globally Synchronized Writes to the PWM1En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.													
1:0	ENUPD0	R/W	0	PWM0 Enable Update Mode										
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Immediate Writes to the PWM0En bit in the PWMENABLE register are used by the PWM generator module immediately.</td></tr> <tr> <td>0x1</td><td>Reserved</td></tr> <tr> <td>0x2</td><td>Locally Synchronized Writes to the PWM0En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0.</td></tr> <tr> <td>0x3</td><td>Globally Synchronized Writes to the PWM0En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.</td></tr> </tbody> </table>	Value	Description	0x0	Immediate Writes to the PWM0En bit in the PWMENABLE register are used by the PWM generator module immediately.	0x1	Reserved	0x2	Locally Synchronized Writes to the PWM0En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0.	0x3	Globally Synchronized Writes to the PWM0En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.
Value	Description													
0x0	Immediate Writes to the PWM0En bit in the PWMENABLE register are used by the PWM generator module immediately.													
0x1	Reserved													
0x2	Locally Synchronized Writes to the PWM0En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0.													
0x3	Globally Synchronized Writes to the PWM0En bit in the PWMENABLE register are used by the PWM generator module the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (PWMCTL) register.													

Register 12: PWM0 Control (PWM0CTL), offset 0x040**Register 13: PWM1 Control (PWM1CTL), offset 0x080****Register 14: PWM2 Control (PWM2CTL), offset 0x0C0**

These registers configure the PWM signal generation blocks (PWM0CTL controls the PWM generator 0 block, and so on). The Register Update mode, Debug mode, Counting mode, and Block Enable mode are all controlled via these registers. The blocks produce the PWM signals, which can be either two independent PWM signals (from the same counter), or a paired set of PWM signals with dead-band delays added.

The PWM0 block produces the PWM0 and PWM1 outputs, the PWM1 block produces the PWM2 and PWM3 outputs, and the PWM2 block produces the PWM4 and PWM5 outputs.

PWM0 Control (PWM0CTL)

Base 0x4002.8000
Offset 0x040
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	DBFALLUPD	DBRISEUPD	DBCTLUFD	GENBUPD	GENAUPD	CMPBUPD	CMPAUPD	LOADUPD	DEBUG	MODE	ENABLE					
Reset	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Bit/Field Name Type Reset Description

31:19 reserved RO 0x000 Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

18 LATCH R/W 0 Latch Fault Input

Value Description

- | | |
|---|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0 | Fault Condition Not Latched
A fault condition is in effect for as long as the generating source is asserting. |
| 1 | Fault Condition Latched
A fault condition is set as the result of the assertion of the faulting source and is held (latched) while the PWMISC INTFAULTn bit is set. Clearing the INTFAULTn bit clears the fault condition. |

Bit/Field	Name	Type	Reset	Description										
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 MINFLTPER bit is set. If a detected fault is in the process of being extended when the MINFLTPER bit is cleared, the fault condition extension is aborted.</p> <p>The delay time is specified by the PWMnMINFLTPER register MFP 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 $\text{PWMClock} * (\text{MFP value} + 1)$ or $\text{PWMClock} * (\text{MFP value} + 2)$.</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 FLTSRC field.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The FAULT input deassertion is unaffected.</td></tr> <tr> <td>1</td><td>The PWMnMINFLTPER one-shot counter is active and extends the period of the fault condition to a minimum period.</td></tr> </tbody> </table>	Value	Description	0	The FAULT input deassertion is unaffected.	1	The PWMnMINFLTPER one-shot counter is active and extends the period of the fault condition to a minimum period.				
Value	Description													
0	The FAULT input deassertion is unaffected.													
1	The PWMnMINFLTPER one-shot counter is active and extends the period of the fault condition to a minimum period.													
16	FLTSRC	R/W	0	<p>Fault Condition Source</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>The Fault condition is determined by the <code>Fault0</code> input.</td></tr> <tr> <td>1</td><td>The Fault condition is determined by the configuration of the PWMnFLTSRC0 and PWMnFLTSRC1 registers.</td></tr> </tbody> </table>	Value	Description	0	The Fault condition is determined by the <code>Fault0</code> input.	1	The Fault condition is determined by the configuration of the PWMnFLTSRC0 and PWMnFLTSRC1 registers.				
Value	Description													
0	The Fault condition is determined by the <code>Fault0</code> input.													
1	The Fault condition is determined by the configuration of the PWMnFLTSRC0 and PWMnFLTSRC1 registers.													
15:14	DBFALLUPD	R/W	0x0	<p>PWMnDBFALL Update Mode</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Immediate The PWMnDBFALL register value is immediately updated on a write.</td></tr> <tr> <td>0x1</td><td>Reserved</td></tr> <tr> <td>0x2</td><td>Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0.</td></tr> <tr> <td>0x3</td><td>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 PWMCTL register.</td></tr> </tbody> </table>	Value	Description	0x0	Immediate The PWMnDBFALL register value is immediately updated on a write.	0x1	Reserved	0x2	Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0.	0x3	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 PWMCTL register.
Value	Description													
0x0	Immediate The PWMnDBFALL register value is immediately updated on a write.													
0x1	Reserved													
0x2	Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0.													
0x3	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 PWMCTL register.													

Bit/Field	Name	Type	Reset	Description										
13:12	DBRISEUPD	R/W	0x0	PWMnDBRISE Update Mode										
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Immediate The PWMnDBRISE register value is immediately updated on a write.</td></tr> <tr> <td>0x1</td><td>Reserved</td></tr> <tr> <td>0x2</td><td>Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0.</td></tr> <tr> <td>0x3</td><td>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 PWMCTL register.</td></tr> </tbody> </table>	Value	Description	0x0	Immediate The PWMnDBRISE register value is immediately updated on a write.	0x1	Reserved	0x2	Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0.	0x3	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 PWMCTL register.
Value	Description													
0x0	Immediate The PWMnDBRISE register value is immediately updated on a write.													
0x1	Reserved													
0x2	Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0.													
0x3	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 PWMCTL register.													
11:10	DBCTLUPD	R/W	0x0	PWMnDBCTL Update Mode										
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Immediate The PWMnDBCTL register value is immediately updated on a write.</td></tr> <tr> <td>0x1</td><td>Reserved</td></tr> <tr> <td>0x2</td><td>Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0.</td></tr> <tr> <td>0x3</td><td>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 PWMCTL register.</td></tr> </tbody> </table>	Value	Description	0x0	Immediate The PWMnDBCTL register value is immediately updated on a write.	0x1	Reserved	0x2	Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0.	0x3	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 PWMCTL register.
Value	Description													
0x0	Immediate The PWMnDBCTL register value is immediately updated on a write.													
0x1	Reserved													
0x2	Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0.													
0x3	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 PWMCTL register.													
9:8	GENBUPD	R/W	0x0	PWMnGENB Update Mode										
				<table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Immediate The PWMnGENB register value is immediately updated on a write.</td></tr> <tr> <td>0x1</td><td>Reserved</td></tr> <tr> <td>0x2</td><td>Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0.</td></tr> <tr> <td>0x3</td><td>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 PWMCTL register.</td></tr> </tbody> </table>	Value	Description	0x0	Immediate The PWMnGENB register value is immediately updated on a write.	0x1	Reserved	0x2	Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0.	0x3	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 PWMCTL register.
Value	Description													
0x0	Immediate The PWMnGENB register value is immediately updated on a write.													
0x1	Reserved													
0x2	Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0.													
0x3	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 PWMCTL register.													

Bit/Field	Name	Type	Reset	Description										
7:6	GENAUPD	R/W	0x0	PWMnGENA Update Mode <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Immediate The PWMnGENA register value is immediately updated on a write.</td></tr> <tr> <td>0x1</td><td>Reserved</td></tr> <tr> <td>0x2</td><td>Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0.</td></tr> <tr> <td>0x3</td><td>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 PWMCTL register.</td></tr> </tbody> </table>	Value	Description	0x0	Immediate The PWMnGENA register value is immediately updated on a write.	0x1	Reserved	0x2	Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0.	0x3	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 PWMCTL register.
Value	Description													
0x0	Immediate The PWMnGENA register value is immediately updated on a write.													
0x1	Reserved													
0x2	Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0.													
0x3	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 PWMCTL register.													
5	CMPBUPD	R/W	0	Comparator B Update Mode <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Locally Synchronized Updates to the PWMnCMPB register are reflected to the generator the next time the counter is 0.</td></tr> <tr> <td>1</td><td>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 PWMCTL register.</td></tr> </tbody> </table>	Value	Description	0	Locally Synchronized Updates to the PWMnCMPB register are reflected to the generator the next time the counter is 0.	1	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 PWMCTL register.				
Value	Description													
0	Locally Synchronized Updates to the PWMnCMPB register are reflected to the generator the next time the counter is 0.													
1	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 PWMCTL register.													
4	CMPAUPD	R/W	0	Comparator A Update Mode <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Locally Synchronized Updates to the PWMnCMPA register are reflected to the generator the next time the counter is 0.</td></tr> <tr> <td>1</td><td>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 PWMCTL register.</td></tr> </tbody> </table>	Value	Description	0	Locally Synchronized Updates to the PWMnCMPA register are reflected to the generator the next time the counter is 0.	1	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 PWMCTL register.				
Value	Description													
0	Locally Synchronized Updates to the PWMnCMPA register are reflected to the generator the next time the counter is 0.													
1	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 PWMCTL register.													
3	LOADUPD	R/W	0	Load Register Update Mode <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0</td><td>Locally Synchronized Updates to the PWMnLOAD register are reflected to the generator the next time the counter is 0.</td></tr> <tr> <td>1</td><td>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 PWMCTL register.</td></tr> </tbody> </table>	Value	Description	0	Locally Synchronized Updates to the PWMnLOAD register are reflected to the generator the next time the counter is 0.	1	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 PWMCTL register.				
Value	Description													
0	Locally Synchronized Updates to the PWMnLOAD register are reflected to the generator the next time the counter is 0.													
1	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 PWMCTL register.													

Bit/Field	Name	Type	Reset	Description
2	DEBUG	R/W	0	Debug Mode
				Value Description
			0	The counter stops running when it next reaches 0 and continues running again when no longer in Debug mode.
			1	The counter always runs when in Debug mode.
1	MODE	R/W	0	Counter Mode
				Value Description
			0	The counter counts down from the load value to 0 and then wraps back to the load value (Count-Down mode).
			1	The counter counts up from 0 to the load value, back down to 0, and then repeats (Count-Up/Down mode).
0	ENABLE	R/W	0	PWM Block Enable
				Value Description
			0	The entire PWM generation block is disabled and not clocked.
			1	The PWM generation block is enabled and produces PWM signals.

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**

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 **PWMnCMPA** register while counting up
- The counter being equal to the **PWMnCMPA** register while counting down
- The counter being equal to the **PWMnCMPB** register while counting up
- The counter being equal to the **PWMnCMPB** 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. The **PWMnRIS** register provides information about which events have caused raw interrupts.

PWM0 Interrupt and Trigger Enable (PWM0INTEN)

Base 0x4002.8000
Offset 0x044
Type R/W, reset 0x0000.0000

																	16
																	17
																	18
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
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
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
reserved																	
Type	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	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
31:14	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	TRCMPBDown	R/W	0	Trigger for Counter= PWMnCMPB Down
	Value	Description		
	1	An ADC trigger pulse is output when the counter matches the value in the PWMnCMPB register value while counting down.		
	0	No ADC trigger is output.		

Bit/Field	Name	Type	Reset	Description
12	TRCMPBU	R/W	0	Trigger for Counter= PWMnCMPB Up Value Description 1 An ADC trigger pulse is output when the counter matches the value in the PWMnCMPB register value while counting up. 0 No ADC trigger is output.
11	TRCMPAD	R/W	0	Trigger for Counter= PWMnCMPA Down Value Description 1 An ADC trigger pulse is output when the counter matches the value in the PWMnCMPA register value while counting down. 0 No ADC trigger is output.
10	TRCMPAU	R/W	0	Trigger for Counter= PWMnCMPA Up Value Description 1 An ADC trigger pulse is output when the counter matches the value in the PWMnCMPA register value while counting up. 0 No ADC trigger is output.
9	TRCNTLOAD	R/W	0	Trigger for Counter= PWMnLOAD Value Description 1 An ADC trigger pulse is output when the counter matches the PWMnLOAD register. 0 No ADC trigger is output.
8	TRCNTZERO	R/W	0	Trigger for Counter=0 Value Description 1 An ADC trigger pulse is output when the counter is 0. 0 No ADC trigger is output.
7:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	INTCMPBD	R/W	0	Interrupt for Counter= PWMnCMPB Down Value Description 1 A raw interrupt occurs when the counter matches the value in the PWMnCMPB register value while counting down. 0 No interrupt.

Bit/Field	Name	Type	Reset	Description
4	INTCMPBU	R/W	0	Interrupt for Counter= PWMnCMPB Up Value Description 1 A raw interrupt occurs when the counter matches the value in the PWMnCMPB register value while counting up. 0 No interrupt.
3	INTCMPAD	R/W	0	Interrupt for Counter= PWMnCMPA Down Value Description 1 A raw interrupt occurs when the counter matches the value in the PWMnCMPA register value while counting down. 0 No interrupt.
2	INTCMPAU	R/W	0	Interrupt for Counter= PWMnCMPA Up Value Description 1 A raw interrupt occurs when the counter matches the value in the PWMnCMPA register value while counting up. 0 No interrupt.
1	INTCNTLOAD	R/W	0	Interrupt for Counter= PWMnLOAD Value Description 1 A raw interrupt occurs when the counter matches the value in the PWMnLOAD register value. 0 No interrupt.
0	INTCNTZERO	R/W	0	Interrupt for Counter=0 Value Description 1 A raw interrupt occurs when the counter is zero. 0 No interrupt.

Register 18: PWM0 Raw Interrupt Status (PWM0RIS), offset 0x048**Register 19: PWM1 Raw Interrupt Status (PWM1RIS), offset 0x088****Register 20: PWM2 Raw Interrupt Status (PWM2RIS), offset 0x0C8**

These registers provide the current set of interrupt sources that are asserted, regardless of whether they cause an interrupt to be asserted to the controller (**PWM0RIS** controls the PWM generator 0 block, and so on). If a bit is set, the event has occurred; if a bit is clear, the event has not occurred. Bits in this register are cleared by writing a 1 to the corresponding bit in the **PWMnISC** register.

PWM0 Raw Interrupt Status (PWM0RIS)

Base 0x4002.8000

Offset 0x048

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	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.
5	INTCMPBD	RO	0	Comparator B Down Interrupt Status
	Value	Description		
	1	The counter has matched the value in the PWMnCMPB register while counting down.		
	0	An interrupt has not occurred.		
	This bit is cleared by writing a 1 to the INTCMPBD bit in the PWMnISC register.			
4	INTCMPBU	RO	0	Comparator B Up Interrupt Status
	Value	Description		
	1	The counter has matched the value in the PWMnCMPB register while counting up.		
	0	An interrupt has not occurred.		
	This bit is cleared by writing a 1 to the INTCMPBU bit in the PWMnISC register.			

Bit/Field	Name	Type	Reset	Description
3	INTCMPAD	RO	0	<p>Comparator A Down Interrupt Status</p> <p>Value Description</p> <p>1 The counter has matched the value in the PWMnCMPA register while counting down.</p> <p>0 An interrupt has not occurred.</p> <p>This bit is cleared by writing a 1 to the INTCMPAD bit in the PWMnISC register.</p>
2	INTCMPOU	RO	0	<p>Comparator A Up Interrupt Status</p> <p>Value Description</p> <p>1 The counter has matched the value in the PWMnCMPA register while counting up.</p> <p>0 An interrupt has not occurred.</p> <p>This bit is cleared by writing a 1 to the INTCMPOU bit in the PWMnISC register.</p>
1	INTCNTLOAD	RO	0	<p>Counter=Load Interrupt Status</p> <p>Value Description</p> <p>1 The counter has matched the value in the PWMnLOAD register.</p> <p>0 An interrupt has not occurred.</p> <p>This bit is cleared by writing a 1 to the INTCNTLOAD bit in the PWMnISC register.</p>
0	INTCNTZERO	RO	0	<p>Counter=0 Interrupt Status</p> <p>Value Description</p> <p>1 The counter has matched zero.</p> <p>0 An interrupt has not occurred.</p> <p>This bit is cleared by writing a 1 to the INTCNTZERO bit in the PWMnISC register.</p>

Register 21: PWM0 Interrupt Status and Clear (PWM0ISC), offset 0x04C**Register 22: PWM1 Interrupt Status and Clear (PWM1ISC), offset 0x08C****Register 23: PWM2 Interrupt Status and Clear (PWM2ISC), offset 0x0CC**

These registers provide the current set of interrupt sources that are asserted to the interrupt controller (**PWM0ISC** controls the PWM generator 0 block, and so on). A bit is set if the event has occurred and is enabled in the **PWMnINTEN** register; if a bit is clear, the event has not occurred or is not enabled. 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
Type	RO	RO	RO	RO	RO	RO										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Type	RO	R/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
reserved																
Type	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	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	INTCMPBD	R/W1C	0	Comparator B Down Interrupt
		Value	Description	
		1	The INTCMPBD bits in the PWMnRIS and PWMnINTEN registers are set, providing an interrupt to the interrupt controller.	
		0	No interrupt has occurred or the interrupt is masked.	
		This bit is cleared by writing a 1. Clearing this bit also clears the INTCMPBD bit in the PWMnRIS register.		
4	INTCMPBU	R/W1C	0	Comparator B Up Interrupt
		Value	Description	
		1	The INTCMPBU bits in the PWMnRIS and PWMnINTEN registers are set, providing an interrupt to the interrupt controller.	
		0	No interrupt has occurred or the interrupt is masked.	
		This bit is cleared by writing a 1. Clearing this bit also clears the INTCMPBU bit in the PWMnRIS register.		

Bit/Field	Name	Type	Reset	Description
3	INTCMPAD	R/W1C	0	<p>Comparator A Down Interrupt</p> <p>Value Description</p> <p>1 The INTCMPAD bits in the PWMnRIS and PWMnINTEN registers are set, providing an interrupt to the interrupt controller.</p> <p>0 No interrupt has occurred or the interrupt is masked.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INTCMPAD bit in the PWMnRIS register.</p>
2	INTCMPAU	R/W1C	0	<p>Comparator A Up Interrupt</p> <p>Value Description</p> <p>1 The INTCMPAU bits in the PWMnRIS and PWMnINTEN registers are set, providing an interrupt to the interrupt controller.</p> <p>0 No interrupt has occurred or the interrupt is masked.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INTCMPAU bit in the PWMnRIS register.</p>
1	INTCNTLOAD	R/W1C	0	<p>Counter=Load Interrupt</p> <p>Value Description</p> <p>1 The INTCNTLOAD bits in the PWMnRIS and PWMnINTEN registers are set, providing an interrupt to the interrupt controller.</p> <p>0 No interrupt has occurred or the interrupt is masked.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INTCNTLOAD bit in the PWMnRIS register.</p>
0	INTCNTZERO	R/W1C	0	<p>Counter=0 Interrupt</p> <p>Value Description</p> <p>1 The INTCNTZERO bits in the PWMnRIS and PWMnINTEN registers are set, providing an interrupt to the interrupt controller.</p> <p>0 No interrupt has occurred or the interrupt is masked.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INTCNTZERO bit in the PWMnRIS register.</p>

Register 24: PWM0 Load (PWM0LOAD), offset 0x050**Register 25: PWM1 Load (PWM1LOAD), offset 0x090****Register 26: PWM2 Load (PWM2LOAD), offset 0x0D0**

These registers contain the load value for the PWM counter (**PWM0LOAD** controls the PWM generator 0 block, and so on). Based on the counter mode configured by the **MODE** bit in the **PWMnCTL** register, this value is either loaded into the counter after it reaches zero or is the limit of up-counting after which the counter decrements back to zero. When this value matches the counter, a pulse is output which can be configured to drive the generation of the **pwmA** and/or **pwmB** signal (via the **PWMnGENA/PWMnGENB** register) or drive an interruptor ADC trigger (via the **PWMnINTEN** register).

If the Load Value Update mode is locally synchronized (based on the **LOADUPD** field encoding in the **PWMnCTL** register), the 16-bit **LOAD** value is used the next time the counter reaches zero. If the update mode is globally synchronized, 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 1079). 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

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

Bit/Field	Name	Type	Reset	Description
31:16	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	LOAD	R/W	0x0000	Counter Load Value The counter load value.

Register 27: PWM0 Counter (PWM0COUNT), offset 0x054**Register 28: PWM1 Counter (PWM1COUNT), offset 0x094****Register 29: PWM2 Counter (PWM2COUNT), offset 0x0D4**

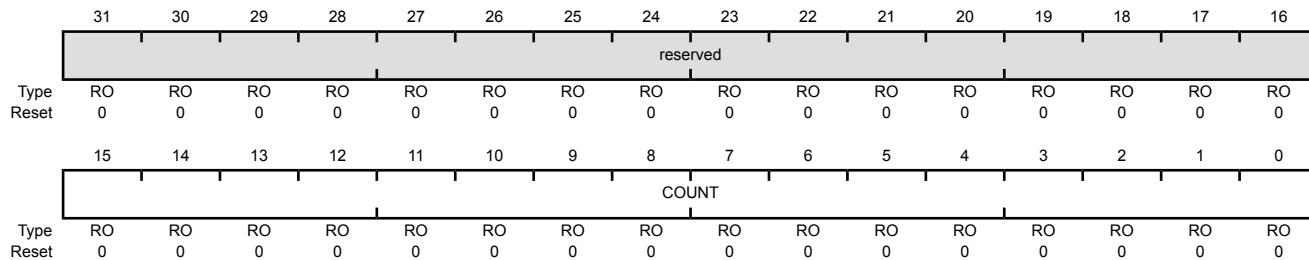
These registers contain the current value of the PWM counter (**PWM0COUNT** is the value of the PWM generator 0 block, and so on). When this value matches zero or the value in the **PWMnLOAD**, **PWMnCMPA**, or **PWMnCMPB** registers, a pulse is output which can be configured to drive the generation of a PWM signal or drive an interrupt or ADC trigger.

PWM0 Counter (PWM0COUNT)

Base 0x4002.8000

Offset 0x054

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	COUNT	RO	0x0000	Counter Value The current value of the counter.

Register 30: PWM0 Compare A (PWM0CMPA), offset 0x058**Register 31: PWM1 Compare A (PWM1CMPA), offset 0x098****Register 32: PWM2 Compare A (PWM2CMPA), offset 0x0D8**

These registers contain a value to be compared against the counter (**PWM0CMPA** controls the PWM generator 0 block, and so on). When this value matches the counter, a pulse is output which can be configured to drive the generation of the pwmA and pwmB signals (via the **PWMnGENA** and **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 1112), then no pulse is ever output.

If the comparator A update mode is locally synchronized (based on the **CMPAUPD** bit in the **PWMnCTL** register), the 16-bit **COMPA** value is used the next time the counter reaches zero. If the update mode is globally synchronized, 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 1079). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWM0 Compare A (PWM0CMPA)

Base 0x4002.8000
Offset 0x058
Type R/W, reset 0x0000.0000

reserved															
Type	RO														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
COMPA															
Type	R/W														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	COMPA	R/W	0x00	Comparator A Value The value to be compared against the counter.

Register 33: PWM0 Compare B (PWM0CMPB), offset 0x05C**Register 34: PWM1 Compare B (PWM1CMPB), offset 0x09C****Register 35: PWM2 Compare B (PWM2CMPB), offset 0x0DC**

These registers contain a value to be compared against the counter (**PWM0CMPB** controls the PWM generator 0 block, and so on). When this value matches the counter, a pulse is output which can be configured to drive the generation of the pwmA and pwmB signals (via the **PWMnGENA** and **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 locally synchronized (based on the **CMPBUPD** bit in the **PWMnCTL** register), the 16-bit **COMPB** value is used the next time the counter reaches zero. If the update mode is globally synchronized, 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 1079). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWM0 Compare B (PWM0CMPB)

Base 0x4002.8000
Offset 0x05C
Type R/W, reset 0x0000.0000

reserved															
Type	RO														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
COMPB															
Type	R/W														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	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	COMPB	R/W	0x0000	Comparator B Value The value to be compared against the counter.

Register 36: PWM0 Generator A Control (PWM0GENA), offset 0x060**Register 37: PWM1 Generator A Control (PWM1GENA), offset 0x0A0****Register 38: PWM2 Generator A Control (PWM2GENA), offset 0x0E0**

These registers control the generation of the pwmA 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 resulting PWM signal.

The **PWM0GENA** register controls generation of the pwm0A signal; **PWM1GENA**, the pwm1A signal; and **PWM2GENA**, the pwm2A signal.

If a zero or load event coincides with a compare A or compare B event, the zero or load action is taken and the compare A or compare B action is ignored. If a compare A event coincides with a compare B event, the compare A action is taken and the compare B action is ignored.

If the Generator A update mode is immediate (based on the GENAUPD field encoding in the **PWMnCTL** register), the ACTCMPBD, ACTCMPBU, ACTCPAD, ACTCPAU, ACTLOAD, and ACTZERO values are used immediately. If the update mode is locally synchronized, these values are used the next time the counter reaches zero. If the update mode is globally synchronized, these values are used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 1079). 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

Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type	RO	RO	RO	RO	R/W										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved															
ACTCMPBD															
ACTCMPBU															
ACTCPAD															
ACTCPAU															
ACTLOAD															
ACTZERO															

Bit/Field	Name	Type	Reset	Description
31:12	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.
11:10	ACTCMPBD	R/W	0x0	Action for Comparator B Down This field specifies the action to be taken when the counter matches comparator B while counting down.
				Value Description
				0x0 Do nothing.
				0x1 Invert pwmA.
				0x2 Drive pwmA Low.
				0x3 Drive pwmA High.

Bit/Field	Name	Type	Reset	Description										
9:8	ACTCMPBU	R/W	0x0	<p>Action for Comparator B Up</p> <p>This field specifies the action to be taken when the counter matches comparator B while counting up. This action can only occur when the MODE bit in the PWMnCTL register is set.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Do nothing.</td></tr> <tr> <td>0x1</td><td>Invert pwmA.</td></tr> <tr> <td>0x2</td><td>Drive pwmA Low.</td></tr> <tr> <td>0x3</td><td>Drive pwmA High.</td></tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmA.	0x2	Drive pwmA Low.	0x3	Drive pwmA High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmA.													
0x2	Drive pwmA Low.													
0x3	Drive pwmA High.													
7:6	ACTCMPAD	R/W	0x0	<p>Action for Comparator A Down</p> <p>This field specifies the action to be taken when the counter matches comparator A while counting down.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Do nothing.</td></tr> <tr> <td>0x1</td><td>Invert pwmA.</td></tr> <tr> <td>0x2</td><td>Drive pwmA Low.</td></tr> <tr> <td>0x3</td><td>Drive pwmA High.</td></tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmA.	0x2	Drive pwmA Low.	0x3	Drive pwmA High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmA.													
0x2	Drive pwmA Low.													
0x3	Drive pwmA High.													
5:4	ACTCMPAU	R/W	0x0	<p>Action for Comparator A Up</p> <p>This field specifies the action to be taken when the counter matches comparator A while counting up. This action can only occur when the MODE bit in the PWMnCTL register is set.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Do nothing.</td></tr> <tr> <td>0x1</td><td>Invert pwmA.</td></tr> <tr> <td>0x2</td><td>Drive pwmA Low.</td></tr> <tr> <td>0x3</td><td>Drive pwmA High.</td></tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmA.	0x2	Drive pwmA Low.	0x3	Drive pwmA High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmA.													
0x2	Drive pwmA Low.													
0x3	Drive pwmA High.													
3:2	ACTLOAD	R/W	0x0	<p>Action for Counter=LOAD</p> <p>This field specifies the action to be taken when the counter matches the value in the PWMnLOAD register.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Do nothing.</td></tr> <tr> <td>0x1</td><td>Invert pwmA.</td></tr> <tr> <td>0x2</td><td>Drive pwmA Low.</td></tr> <tr> <td>0x3</td><td>Drive pwmA High.</td></tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmA.	0x2	Drive pwmA Low.	0x3	Drive pwmA High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmA.													
0x2	Drive pwmA Low.													
0x3	Drive pwmA High.													

Bit/Field	Name	Type	Reset	Description
1:0	ACTZERO	R/W	0x0	Action for Counter=0 This field specifies the action to be taken when the counter is zero.
Value Description				
0x0 Do nothing.				
0x1 Invert pwmA.				
0x2 Drive pwmA Low.				
0x3 Drive pwmA High.				

Register 39: PWM0 Generator B Control (PWM0GENB), offset 0x064**Register 40: PWM1 Generator B Control (PWM1GENB), offset 0x0A4****Register 41: PWM2 Generator B Control (PWM2GENB), offset 0x0E4**

These registers control the generation of the pwmB 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 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 resulting PWM signal.

The **PWM0GENB** register controls generation of the pwm0B signal; **PWM1GENB**, the pwm1B signal; and **PWM2GENB**, the pwm2B signal.

If a zero or load event coincides with a compare A or compare B event, the zero or load action is taken and the compare A or compare B action is ignored. If a compare A event coincides with a compare B event, the compare B action is taken and the compare A action is ignored.

If the Generator B update mode is immediate (based on the GENBUPD field encoding in the **PWMnCTL** register), the ACTCMPBD, ACTCMPBU, ACTCMPAD, ACTCMPPAU, ACTLOAD, and ACTZERO values are used immediately. If the update mode is locally synchronized, these values are used the next time the counter reaches zero. If the update mode is globally synchronized, these values are used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 1079). 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

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

Bit/Field	Name	Type	Reset	Description										
9:8	ACTCMPBU	R/W	0x0	<p>Action for Comparator B Up</p> <p>This field specifies the action to be taken when the counter matches comparator B while counting up. This action can only occur when the MODE bit in the PWMnCTL register is set.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Do nothing.</td></tr> <tr> <td>0x1</td><td>Invert pwmB.</td></tr> <tr> <td>0x2</td><td>Drive pwmB Low.</td></tr> <tr> <td>0x3</td><td>Drive pwmB High.</td></tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmB.	0x2	Drive pwmB Low.	0x3	Drive pwmB High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmB.													
0x2	Drive pwmB Low.													
0x3	Drive pwmB High.													
7:6	ACTCMPAD	R/W	0x0	<p>Action for Comparator A Down</p> <p>This field specifies the action to be taken when the counter matches comparator A while counting down.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Do nothing.</td></tr> <tr> <td>0x1</td><td>Invert pwmB.</td></tr> <tr> <td>0x2</td><td>Drive pwmB Low.</td></tr> <tr> <td>0x3</td><td>Drive pwmB High.</td></tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmB.	0x2	Drive pwmB Low.	0x3	Drive pwmB High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmB.													
0x2	Drive pwmB Low.													
0x3	Drive pwmB High.													
5:4	ACTCMPAU	R/W	0x0	<p>Action for Comparator A Up</p> <p>This field specifies the action to be taken when the counter matches comparator A while counting up. This action can only occur when the MODE bit in the PWMnCTL register is set.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Do nothing.</td></tr> <tr> <td>0x1</td><td>Invert pwmB.</td></tr> <tr> <td>0x2</td><td>Drive pwmB Low.</td></tr> <tr> <td>0x3</td><td>Drive pwmB High.</td></tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmB.	0x2	Drive pwmB Low.	0x3	Drive pwmB High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmB.													
0x2	Drive pwmB Low.													
0x3	Drive pwmB High.													
3:2	ACTLOAD	R/W	0x0	<p>Action for Counter=LOAD</p> <p>This field specifies the action to be taken when the counter matches the load value.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Do nothing.</td></tr> <tr> <td>0x1</td><td>Invert pwmB.</td></tr> <tr> <td>0x2</td><td>Drive pwmB Low.</td></tr> <tr> <td>0x3</td><td>Drive pwmB High.</td></tr> </tbody> </table>	Value	Description	0x0	Do nothing.	0x1	Invert pwmB.	0x2	Drive pwmB Low.	0x3	Drive pwmB High.
Value	Description													
0x0	Do nothing.													
0x1	Invert pwmB.													
0x2	Drive pwmB Low.													
0x3	Drive pwmB High.													

Bit/Field	Name	Type	Reset	Description
1:0	ACTZERO	R/W	0x0	Action for Counter=0 This field specifies the action to be taken when the counter is 0.
Value Description				
0x0 Do nothing.				
0x1 Invert pwmB.				
0x2 Drive pwmB Low.				
0x3 Drive pwmB High.				

Register 42: PWM0 Dead-Band Control (PWM0DBCTL), offset 0x068**Register 43: PWM1 Dead-Band Control (PWM1DBCTL), offset 0x0A8****Register 44: PWM2 Dead-Band Control (PWM2DBCTL), offset 0x0E8**

The **PWMnDBCTL** register controls the dead-band generator, which produces the **PWMn** signals based on the **pwmA** and **pwmB** signals. When disabled, the **pwmA** signal passes through to the **pwmA'** signal and the **pwmB** signal passes through to the **pwmB'** signal. When dead-band control is enabled, the **pwmB** signal is ignored, the **pwmA'** signal is generated by delaying the rising edge(s) of the **pwmA** signal by the value in the **PWMnDBRISE** register (see page 1123), and the **pwmB'** signal is generated by inverting the **pwmA** signal and delaying the falling edge(s) of the **pwmA** signal by the value in the **PWMnDBFALL** register (see page 1124). The Output Control block outputs the **pwm0A'** signal on the **PWM0** signal and the **pwm0B'** signal on the **PWM1** signal. In a similar manner, **PWM2** and **PWM3** are produced from the **pwm1A'** and **pwm1B'** signals, and **PWM4** and **PWM5** are produced from the **pwm2A'** and **pwm2B'** signals.

If the Dead-Band Control mode is immediate (based on the **DBCTLUPD** field encoding in the **PWMnCTL** register), the **ENABLE** bit value is used immediately. If the update mode is locally synchronized, this value is used the next time the counter reaches zero. If the update mode is globally synchronized, this value 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 1079). 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

reserved															
Type	RO														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved															
Type	RO	R/W													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ENABLE															

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	ENABLE	R/W	0	Dead-Band Generator Enable
				Value Description
				1 The dead-band generator modifies the pwmA signal by inserting dead bands into the pwmA' and pwmB' signals.
				0 The pwmA and pwmB signals pass through to the pwmA' and pwmB' signals unmodified.

Register 45: PWM0 Dead-Band Rising-Edge Delay (PWM0DBRISE), offset 0x06C

Register 46: PWM1 Dead-Band Rising-Edge Delay (PWM1DBRISE), offset 0x0AC

Register 47: PWM2 Dead-Band Rising-Edge Delay (PWM2DBRISE), offset 0x0EC

The **PWMnDBRISE** register contains the number of clock cycles to delay the rising edge of the pwmA signal when generating the pwmA' signal. If the dead-band generator is disabled through the **PWMnDBCTL** register, this register is ignored. If the value of this register is larger than the width of a High pulse on the pwmA 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 pwmA High time always exceeds the rising-edge delay.

If the Dead-Band Rising-Edge Delay mode is immediate (based on the DBRISEUPD field encoding in the **PWMnCTL** register), the 12-bit RISEDELAY value is used immediately. If the update mode is locally synchronized, this value is used the next time the counter reaches zero. If the update mode is globally synchronized, this value 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 1079). 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	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.
11:0	RISEDELAY	R/W	0x000	Dead-Band Rise Delay The number of clock cycles to delay the rising edge of pwmA' after the rising edge of pwmA.

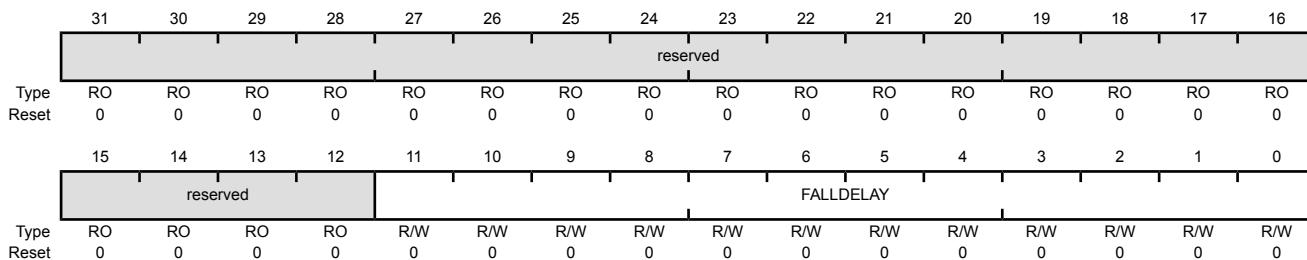
Register 48: PWM0 Dead-Band Falling-Edge-Delay (PWM0DBFALL), offset 0x070**Register 49: PWM1 Dead-Band Falling-Edge-Delay (PWM1DBFALL), offset 0x0B0****Register 50: PWM2 Dead-Band Falling-Edge-Delay (PWM2DBFALL), offset 0x0F0**

The **PWMnDBFALL** register contains the number of clock cycles to delay the rising edge of the pwmB' signal from the falling edge of the pwmA signal. If the dead-band generator is disabled through the **PWMnDBCTL** register, this register is ignored. If the value of this register is larger than the width of a Low pulse on the pwmA 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 pwmA Low time always exceeds the falling-edge delay.

If the Dead-Band Falling-Edge-Delay mode is immediate (based on the DBFALLUP field encoding in the **PWMnCTL** register), the 12-bit FALDELAY value is used immediately. If the update mode is locally synchronized, this value is used the next time the counter reaches zero. If the update mode is globally synchronized, this value 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 1079). 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



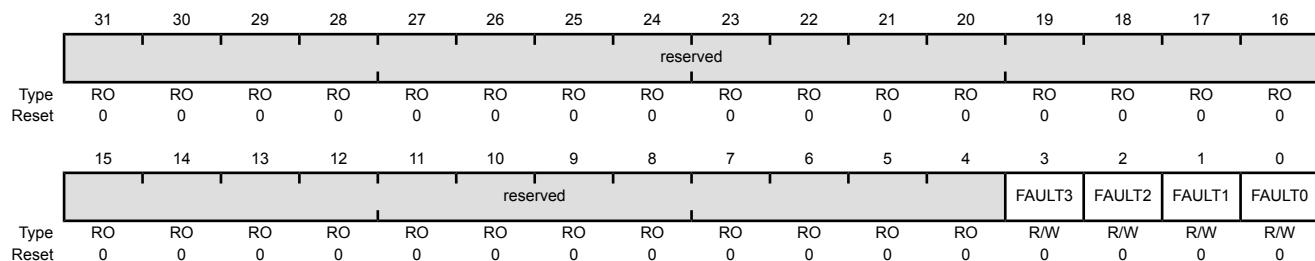
Register 51: PWM0 Fault Source 0 (PWM0FLTSRC0), offset 0x074**Register 52: PWM1 Fault Source 0 (PWM1FLTSRC0), offset 0x0B4****Register 53: PWM2 Fault Source 0 (PWM2FLTSRC0), offset 0x0F4**

This register specifies which fault pin inputs are used to generate 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 **PWM_nFLTSRC0** portion of the fault condition. The **PWM_nFLTSRC0** fault condition is then ORed with the **PWM_nFLTSRC1** fault condition to generate the final fault condition for the PWM generator.

If the FLTSRC bit in the **PWM_nCTL** register (see page 1100) is clear, only the Fault₀ signal affects the fault condition generated. Otherwise, sources defined in **PWM_nFLTSRC0** and **PWM_nFLTSRC1** affect the fault condition generated.

PWM0 Fault Source 0 (PWM0FLTSRC0)

Base 0x4002.8000
Offset 0x074
Type R/W, reset 0x0000.0000



Bit/Field Name Type Reset Description

31:4 reserved RO 0x0000 Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

3 FAULT3 R/W 0 Fault3 Input

Value	Description
0	The Fault3 signal is suppressed and cannot generate a fault condition.
1	The Fault3 signal value is ORed with all other fault condition generation inputs (Fault _n signals and digital comparators).

Note: The FLTSRC bit in the **PWM_nCTL** register must be set for this bit to affect fault condition generation.

2 FAULT2 R/W 0 Fault2 Input

Value	Description
0	The Fault2 signal is suppressed and cannot generate a fault condition.
1	The Fault2 signal value is ORed with all other fault condition generation inputs (Fault _n signals and digital comparators).

Note: The FLTSRC bit in the **PWM_nCTL** register must be set for this bit to affect fault condition generation.

Bit/Field	Name	Type	Reset	Description
1	FAULT1	R/W	0	<p>Fault1 Input</p> <p>Value Description</p> <p>0 The Fault1 signal is suppressed and cannot generate a fault condition.</p> <p>1 The Fault1 signal value is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).</p> <p>Note: The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation.</p>
0	FAULT0	R/W	0	<p>Fault0 Input</p> <p>Value Description</p> <p>0 The Fault0 signal is suppressed and cannot generate a fault condition.</p> <p>1 The Fault0 signal value is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).</p> <p>Note: The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation.</p>

Register 54: PWM0 Fault Source 1 (PWM0FLTSRC1), offset 0x078**Register 55: PWM1 Fault Source 1 (PWM1FLTSRC1), offset 0x0B8****Register 56: PWM2 Fault Source 1 (PWM2FLTSRC1), offset 0x0F8**

This register specifies which digital comparator triggers from the ADC are used to generate 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 1100) 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

reserved															
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
Type	RO	R/W													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
reserved															
Type	RO	DCMP7	DCMP6	DCMP5	DCMP4	DCMP3	DCMP2	DCMP1	DCMP0						
Reset	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	DCMP7	R/W	0	Digital Comparator 7
		Value		Description
		0		The trigger from digital comparator 7 is suppressed and cannot generate a fault condition.
		1		The trigger from digital comparator 7 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators).

Note: The **FLTSRC** bit in the **PWMnCTL** register must be set for this bit to affect fault condition generation.

Bit/Field	Name	Type	Reset	Description
6	DCMP6	R/W	0	Digital Comparator 6 Value Description 0 The trigger from digital comparator 6 is suppressed and cannot generate a fault condition. 1 The trigger from digital comparator 6 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators). Note: The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation.
5	DCMP5	R/W	0	Digital Comparator 5 Value Description 0 The trigger from digital comparator 5 is suppressed and cannot generate a fault condition. 1 The trigger from digital comparator 5 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators). Note: The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation.
4	DCMP4	R/W	0	Digital Comparator 4 Value Description 0 The trigger from digital comparator 4 is suppressed and cannot generate a fault condition. 1 The trigger from digital comparator 4 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators). Note: The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation.
3	DCMP3	R/W	0	Digital Comparator 3 Value Description 0 The trigger from digital comparator 3 is suppressed and cannot generate a fault condition. 1 The trigger from digital comparator 3 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators). Note: The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation.

Bit/Field	Name	Type	Reset	Description
2	DCMP2	R/W	0	Digital Comparator 2 Value Description 0 The trigger from digital comparator 2 is suppressed and cannot generate a fault condition. 1 The trigger from digital comparator 2 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators). Note: The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation.
1	DCMP1	R/W	0	Digital Comparator 1 Value Description 0 The trigger from digital comparator 1 is suppressed and cannot generate a fault condition. 1 The trigger from digital comparator 1 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators). Note: The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation.
0	DCMP0	R/W	0	Digital Comparator 0 Value Description 0 The trigger from digital comparator 0 is suppressed and cannot generate a fault condition. 1 The trigger from digital comparator 0 is ORed with all other fault condition generation inputs (Faultn signals and digital comparators). Note: The FLTSRC bit in the PWMnCTL register must be set for this bit to affect fault condition generation.

Register 57: PWM0 Minimum Fault Period (PWM0MINFLTPER), offset 0x07C**Register 58: PWM1 Minimum Fault Period (PWM1MINFLTPER), offset 0x0BC****Register 59: PWM2 Minimum Fault Period (PWM2MINFLTPER), offset 0x0FC**

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

reserved															
Type	R/W														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MFP															
Type	RO														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

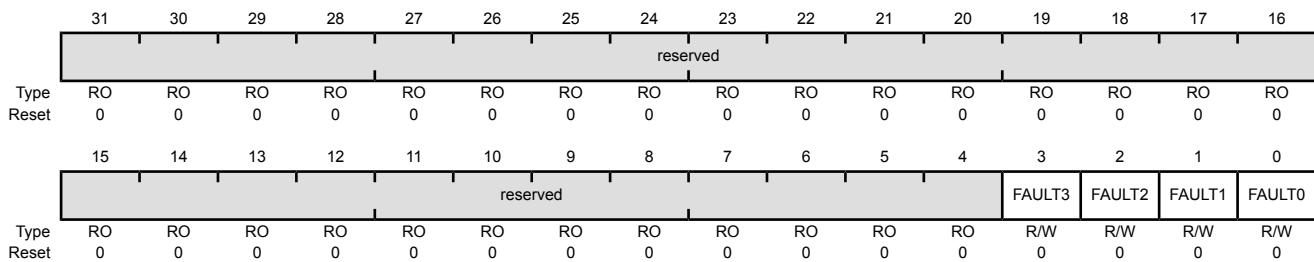
Bit/Field	Name	Type	Reset	Description
31:16	reserved	R/W	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MFP	RO	0x0000	Minimum Fault Period The number of PWM clocks by which a fault condition is extended when the delay is enabled by PWMnCTL MINFLTPER .

Register 60: PWM0 Fault Pin Logic Sense (PWM0FLTSEN), offset 0x800**Register 61: PWM1 Fault Pin Logic Sense (PWM1FLTSEN), offset 0x880****Register 62: PWM2 Fault Pin Logic Sense (PWM2FLTSEN), offset 0x900****Register 63: 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



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	FAULT3	R/W	0	Fault3 Sense Value Description 0 An error is indicated if the Fault3 signal is High. 1 An error is indicated if the Fault3 signal is Low.
2	FAULT2	R/W	0	Fault2 Sense Value Description 0 An error is indicated if the Fault2 signal is High. 1 An error is indicated if the Fault2 signal is Low.
1	FAULT1	R/W	0	Fault1 Sense Value Description 0 An error is indicated if the Fault1 signal is High. 1 An error is indicated if the Fault1 signal is Low.
0	FAULT0	R/W	0	Fault0 Sense Value Description 0 An error is indicated if the Fault0 signal is High. 1 An error is indicated if the Fault0 signal is Low.

Register 64: PWM0 Fault Status 0 (PWM0FLTSTAT0), offset 0x804**Register 65: PWM1 Fault Status 0 (PWM1FLTSTAT0), offset 0x884****Register 66: PWM2 Fault Status 0 (PWM2FLTSTAT0), offset 0x904**

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

																16
																17
																18
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	-	-	-	-
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	reserved	FAULT3	FAULT2	FAULT1	FAULT0											

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	FAULT3	-	0	<p>Fault Input 3</p> <p>If the PWMnCTL register LATCH bit is clear, this bit is RO and represents the current state of the FAULT3 input signal after the logic sense adjustment.</p> <p>If the PWMnCTL register LATCH bit is set, this bit is R/W1C and represents a sticky version of the FAULT3 input signal after the logic sense adjustment.</p> <ul style="list-style-type: none"> ■ If FAULT3 is set, the input transitioned to the active state previously. ■ If FAULT3 is clear, the input has not transitioned to the active state since the last time it was cleared. ■ The FAULT3 bit is cleared by writing it with the value 1.

Bit/Field	Name	Type	Reset	Description
2	FAULT2	-	0	<p>Fault Input 2</p> <p>If the PWMnCTL register LATCH bit is clear, this bit is RO and represents the current state of the FAULT2 input signal after the logic sense adjustment.</p> <p>If the PWMnCTL register LATCH bit is set, this bit is R/W1C and represents a sticky version of the FAULT2 input signal after the logic sense adjustment.</p> <ul style="list-style-type: none"> ■ If FAULT2 is set, the input transitioned to the active state previously. ■ If FAULT2 is clear, the input has not transitioned to the active state since the last time it was cleared. ■ The FAULT2 bit is cleared by writing it with the value 1.
1	FAULT1	-	0	<p>Fault Input 1</p> <p>If the PWMnCTL register LATCH bit is clear, this bit is RO and represents the current state of the FAULT1 input signal after the logic sense adjustment.</p> <p>If the PWMnCTL register LATCH bit is set, this bit is R/W1C and represents a sticky version of the FAULT1 input signal after the logic sense adjustment.</p> <ul style="list-style-type: none"> ■ If FAULT1 is set, the input transitioned to the active state previously. ■ If FAULT1 is clear, the input has not transitioned to the active state since the last time it was cleared. ■ The FAULT1 bit is cleared by writing it with the value 1.
0	FAULT0	-	0	<p>Fault Input 0</p> <p>If the PWMnCTL register LATCH bit is clear, this bit is RO and represents the current state of the FAULT0 input signal after the logic sense adjustment.</p> <p>If the PWMnCTL register LATCH bit is set, this bit is R/W1C and represents a sticky version of the FAULT0 input signal after the logic sense adjustment.</p> <ul style="list-style-type: none"> ■ If FAULT0 is set, the input transitioned to the active state previously. ■ If FAULT0 is clear, the input has not transitioned to the active state since the last time it was cleared. ■ The FAULT0 bit is cleared by writing it with the value 1.

Register 67: PWM0 Fault Status 1 (PWM0FLTSTAT1), offset 0x808**Register 68: PWM1 Fault Status 1 (PWM1FLTSTAT1), offset 0x888****Register 69: PWM2 Fault Status 1 (PWM2FLTSTAT1), offset 0x908**

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

																16
																17
																18
																19
																20
																21
																22
																23
																24
																25
																26
																27
																28
																29
																30
																31
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
Type	RO	-	-	-	-	-	-	-	0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Type	RO	DCMP7	DCMP6	DCMP5	DCMP4	DCMP3	DCMP2	DCMP1	DCMP0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
																reserved
																9
																8
																7
																6
																5
																4
																3
																2
																1
																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	DCMP7	-	0	<p>Digital Comparator 7 Trigger</p> <p>If the PWMnCTL register LATCH bit is clear, this bit represents the current state of the Digital Comparator 7 trigger input.</p> <p>If the PWMnCTL register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> ■ If DCMP7 is set, the trigger transitioned to the active state previously. ■ If DCMP7 is clear, the trigger has not transitioned to the active state since the last time it was cleared. ■ The DCMP7 bit is cleared by writing it with the value 1.

Bit/Field	Name	Type	Reset	Description
6	DCMP6	-	0	<p>Digital Comparator 6 Trigger</p> <p>If the PWMnCTL register LATCH bit is clear, this bit represents the current state of the Digital Comparator 6 trigger input.</p> <p>If the PWMnCTL register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> ■ If DCMP6 is set, the trigger transitioned to the active state previously. ■ If DCMP6 is clear, the trigger has not transitioned to the active state since the last time it was cleared. ■ The DCMP6 bit is cleared by writing it with the value 1.
5	DCMP5	-	0	<p>Digital Comparator 5 Trigger</p> <p>If the PWMnCTL register LATCH bit is clear, this bit represents the current state of the Digital Comparator 5 trigger input.</p> <p>If the PWMnCTL register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> ■ If DCMP5 is set, the trigger transitioned to the active state previously. ■ If DCMP5 is clear, the trigger has not transitioned to the active state since the last time it was cleared. ■ The DCMP5 bit is cleared by writing it with the value 1.
4	DCMP4	-	0	<p>Digital Comparator 4 Trigger</p> <p>If the PWMnCTL register LATCH bit is clear, this bit represents the current state of the Digital Comparator 4 trigger input.</p> <p>If the PWMnCTL register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> ■ If DCMP4 is set, the trigger transitioned to the active state previously. ■ If DCMP4 is clear, the trigger has not transitioned to the active state since the last time it was cleared. ■ The DCMP4 bit is cleared by writing it with the value 1.
3	DCMP3	-	0	<p>Digital Comparator 3 Trigger</p> <p>If the PWMnCTL register LATCH bit is clear, this bit represents the current state of the Digital Comparator 3 trigger input.</p> <p>If the PWMnCTL register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> ■ If DCMP3 is set, the trigger transitioned to the active state previously. ■ If DCMP3 is clear, the trigger has not transitioned to the active state since the last time it was cleared. ■ The DCMP3 bit is cleared by writing it with the value 1.

Bit/Field	Name	Type	Reset	Description
2	DCMP2	-	0	<p>Digital Comparator 2 Trigger</p> <p>If the PWMnCTL register LATCH bit is clear, this bit represents the current state of the Digital Comparator 2 trigger input.</p> <p>If the PWMnCTL register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> ■ If DCMP2 is set, the trigger transitioned to the active state previously. ■ If DCMP2 is clear, the trigger has not transitioned to the active state since the last time it was cleared. ■ The DCMP2 bit is cleared by writing it with the value 1.
1	DCMP1	-	0	<p>Digital Comparator 1 Trigger</p> <p>If the PWMnCTL register LATCH bit is clear, this bit represents the current state of the Digital Comparator 1 trigger input.</p> <p>If the PWMnCTL register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> ■ If DCMP1 is set, the trigger transitioned to the active state previously. ■ If DCMP1 is clear, the trigger has not transitioned to the active state since the last time it was cleared. ■ The DCMP1 bit is cleared by writing it with the value 1.
0	DCMP0	-	0	<p>Digital Comparator 0 Trigger</p> <p>If the PWMnCTL register LATCH bit is clear, this bit represents the current state of the Digital Comparator 0 trigger input.</p> <p>If the PWMnCTL register LATCH bit is set, this bit represents a sticky version of the trigger.</p> <ul style="list-style-type: none"> ■ If DCMP0 is set, the trigger transitioned to the active state previously. ■ If DCMP0 is clear, the trigger has not transitioned to the active state since the last time it was cleared. ■ The DCMP0 bit is cleared by writing it with the value 1.

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 LM3S9L97 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.

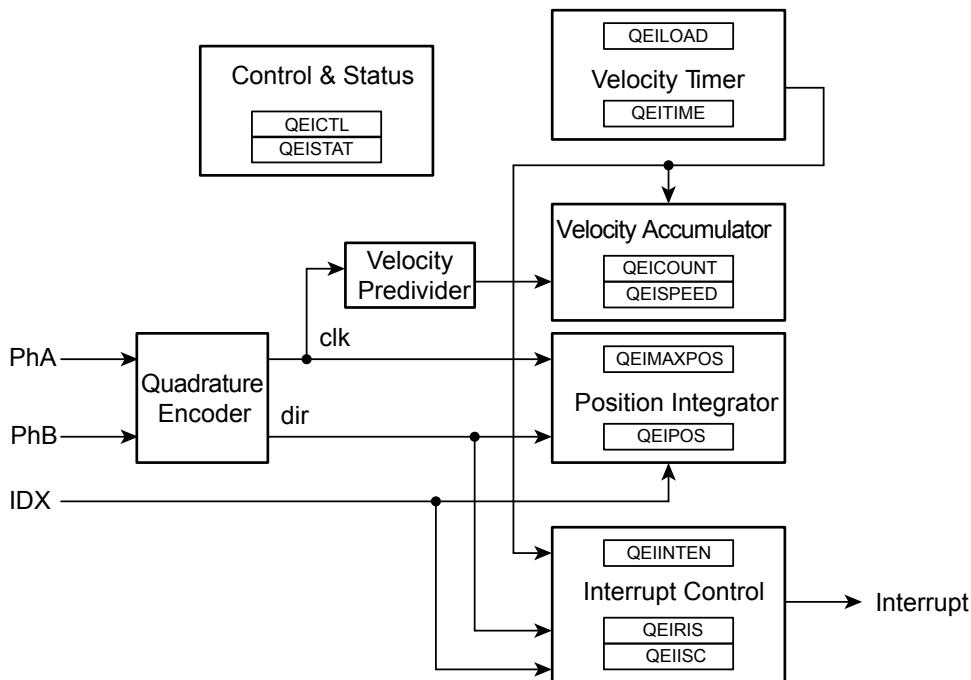
The Stellaris® LM3S9L97 microcontroller includes two QEI modules providing control of two motors at the same time with the following features:

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

22.1 Block Diagram

Figure 22-1 on page 1138 provides a block diagram of a Stellaris QEI module.

Figure 22-1. QEI Block Diagram



22.2 Signal Description

Table 22-1 on page 1138 and Table 22-2 on page 1139 list the external signals of the QEI module and describe the function of each. The QEI signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these QEI signals. The AFSEL bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 447) should be set to choose the QEI function. The number in parentheses is the encoding that must be programmed into the PMCn field in the **GPIO Port Control (GPIOPCTL)** register (page 465) to assign the QEI signal to the specified GPIO port pin. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 423.

Table 22-1. Signals for QEI (100LQFP)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
IDX0	10 72 90 92 100	PD0 (3) PB2 (2) PB6 (5) PB4 (6) PD7 (1)	I	TTL	QEI module 0 index.
IDX1	61 84	PF1 (2) PH2 (1)	I	TTL	QEI module 1 index.
PhA0	11 25 95	PD1 (3) PC4 (2) PE2 (4)	I	TTL	QEI module 0 phase A.
PhA1	96	PE3 (3)	I	TTL	QEI module 1 phase A.

Table 22-1. Signals for QEI (100LQFP) (continued)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
PhB0	22 23 47 83 96	PC7 (2) PC6 (2) 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.

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

Table 22-2. Signals for QEI (108BGA)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
IDX0	G1 A11 A7 A6 A2	PD0 (3) PB2 (2) PB6 (5) PB4 (6) PD7 (1)	I	TTL	QEI module 0 index.
IDX1	H12 D11	PF1 (2) PH2 (1)	I	TTL	QEI module 1 index.
PhA0	G2 L1 A4	PD1 (3) PC4 (2) PE2 (4)	I	TTL	QEI module 0 phase A.
PhA1	B4	PE3 (3)	I	TTL	QEI module 1 phase A.
PhB0	L2 M2 M9 D10 B4	PC7 (2) PC6 (2) PF0 (2) PH3 (1) PE3 (4)	I	TTL	QEI module 0 phase B.
PhB1	G2 C10 A4	PD1 (11) PG7 (1) PE2 (3)	I	TTL	QEI module 1 phase B.

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

22.3 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 a specified number of 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 **QEICTL** register (see page 1144).

When the QEI module is set to use the quadrature phase mode (**SIGMODE** bit is clear), 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** edge 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. The reset mode is determined by the **RESMODE** bit of the **QEICTL** register.

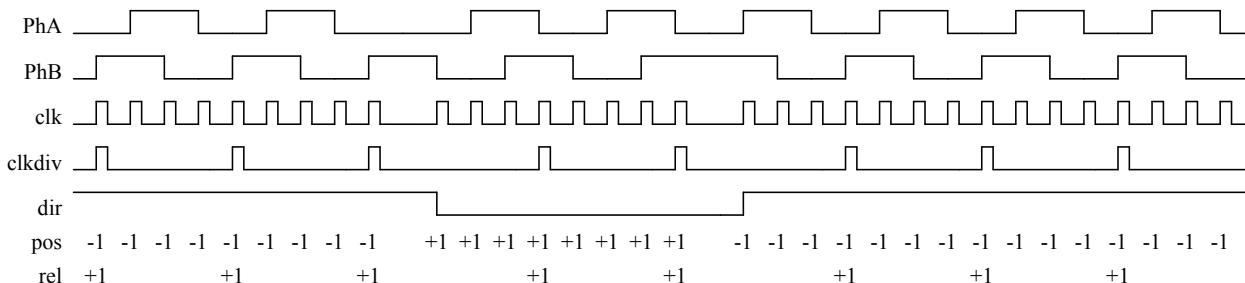
When **RESMODE** is set, the positional counter is reset when the index pulse is sensed. This mode limits the positional counter to the values [0:N-1], where N is the number of phase edges in a full revolution of the encoder wheel. The **QEI Maximum Position (QEIMAXPOS)** register must be programmed 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 clear, 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.

Velocity capture uses a configurable timer and a count register. The timer 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 **QEI Velocity (QEISPEED)** register, while the edge count for the current time period is being accumulated in the **QEI Velocity Counter (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 (overwriting the previous value), the **QEICOUNT** register is cleared, 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 1140 shows how the Stellaris quadrature encoder converts the phase input signals into clock pulses, the direction signal, and how the velocity predivider operates (in Divide by 4 mode).

Figure 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 **QEI Timer Load (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 required to be able to capture enough edges to have a meaningful result. At higher encoder speeds, both a shorter timer period and/or the velocity predivider can be used.

The following equation converts the velocity counter value into an rpm value:

$$\text{rpm} = (\text{clock} * (2 ^ \text{VELDIV}) * \text{SPEED} * 60) \div (\text{LOAD} * \text{ppr} * \text{edges})$$

where:

`clock` is the controller clock rate

`ppr` is the number of pulses per revolution of the physical encoder

`edges` is 2 or 4, based on the capture mode set in the **QEICTL** register (2 for `CAPMODE` clear and 4 for `CAPMODE` set)

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` is clear) 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 because 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 and reduce 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, the load value can be a power of 2. For other encoders, a load value must be selected such that the product is very close to a power of 2. 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 microcontroller'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.

22.4 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 276).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module (see page 285).
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register. To determine which GPIOs to configure, see Table 24-4 on page 1182.
4. Configure the **PMCn** fields in the **GPIOPCTL** register to assign the QEI signals to the appropriate pins (see page 465 and Table 24-5 on page 1190).
5. Configure the quadrature encoder to capture edges on both signals and maintain an absolute position by resetting on index pulses. A 1000-line encoder with four edges per line, results in 4000 pulses per revolution; therefore, set the maximum position to 3999 (0xF9F) as 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.
6. Enable the quadrature encoder by setting bit 0 of the **QEICTL** register.
7. Delay until the encoder position is required.
8. Read the encoder position by reading the **QEI Position (QEIPOS)** register value.

22.5 Register Map

Table 22-3 on page 1142 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 276).

Table 22-3. QEI Register Map

Offset	Name	Type	Reset	Description	See page
0x000	QEICTL	R/W	0x0000.0000	QEI Control	1144
0x004	QEISTAT	RO	0x0000.0000	QEI Status	1147
0x008	QEIPOS	R/W	0x0000.0000	QEI Position	1148
0x00C	QEIMAXPOS	R/W	0x0000.0000	QEI Maximum Position	1149
0x010	QEILOAD	R/W	0x0000.0000	QEI Timer Load	1150
0x014	QEITIME	RO	0x0000.0000	QEI Timer	1151
0x018	QEICOUNT	RO	0x0000.0000	QEI Velocity Counter	1152
0x01C	QEISPEED	RO	0x0000.0000	QEI Velocity	1153

Table 22-3. QEI Register Map (continued)

Offset	Name	Type	Reset	Description	See page
0x020	QEIIINTEN	R/W	0x0000.0000	QEI Interrupt Enable	1154
0x024	QEIRIS	RO	0x0000.0000	QEI Raw Interrupt Status	1156
0x028	QEIIISC	R/W1C	0x0000.0000	QEI Interrupt Status and Clear	1158

22.6 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
Type	reserved		FILTEN	STALLEN	INV1	INV2	INVA	VELDIV			VELEN	RESMODE	CAPMODE	SIGMODE	SWAP	ENABLE
Reset	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

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	<p>Input Filter Prescale Count</p> <p>This field controls the frequency of the input update.</p> <p>When this field is clear, the input is sampled after 2 system clocks. When this field is 0x1, the input is sampled after 3 system clocks. Similarly, when this field is 0xF, the input is sampled after 17 clocks.</p>						
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	<p>Enable Input Filter</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The QEI inputs are not filtered.</td> </tr> <tr> <td>1</td> <td>Enables the digital noise filter on the QEI input signals. Inputs must be stable for 3 consecutive clock edges before the edge detector is updated.</td> </tr> </tbody> </table>	Value	Description	0	The QEI inputs are not filtered.	1	Enables the digital noise filter on the QEI input signals. Inputs must be stable for 3 consecutive clock edges before the edge detector is updated.
Value	Description									
0	The QEI inputs are not filtered.									
1	Enables the digital noise filter on the QEI input signals. Inputs must be stable for 3 consecutive clock edges before the edge detector is updated.									
12	STALLEN	R/W	0	<p>Stall QEI</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The QEI module does not stall when the microcontroller is stopped by a debugger.</td> </tr> <tr> <td>1</td> <td>The QEI module stalls when the microcontroller is stopped by a debugger.</td> </tr> </tbody> </table>	Value	Description	0	The QEI module does not stall when the microcontroller is stopped by a debugger.	1	The QEI module stalls when the microcontroller is stopped by a debugger.
Value	Description									
0	The QEI module does not stall when the microcontroller is stopped by a debugger.									
1	The QEI module stalls when the microcontroller is stopped by a debugger.									

Bit/Field	Name	Type	Reset	Description
11	INVI	R/W	0	Invert Index Pulse Value Description 0 No effect. 1 Inverts the <code>IDX</code> input.
10	INVB	R/W	0	Invert PhB Value Description 0 No effect. 1 Inverts the <code>PhB</code> input.
9	INVA	R/W	0	Invert PhA Value Description 0 No effect. 1 Inverts the <code>PhA</code> input.
8:6	VELDIV	R/W	0x0	Predivide Velocity This field defines the predivider of the input quadrature pulses before being applied to the QEICOUNT accumulator. Value Predivider 0x0 ÷1 0x1 ÷2 0x2 ÷4 0x3 ÷8 0x4 ÷16 0x5 ÷32 0x6 ÷64 0x7 ÷128
5	VELEN	R/W	0	Capture Velocity Value Description 0 No effect. 1 Enables capture of the velocity of the quadrature encoder.
4	RESMODE	R/W	0	Reset Mode Value Description 0 The position counter is reset when it reaches the maximum as defined by the <code>MAXPOS</code> field in the QEIMAXPOS register. 1 The position counter is reset when the index pulse is captured.

Bit/Field	Name	Type	Reset	Description
3	CAPMODE	R/W	0	Capture Mode Value Description 0 Only the PhA edges are counted. 1 The PhA and PhB edges are counted, providing twice the positional resolution but half the range.
2	SIGMODE	R/W	0	Signal Mode Value Description 0 The PhA and PhB signals operate as quadrature phase signals. 1 The PhA and PhB signals operate as clock and direction.
1	SWAP	R/W	0	Swap Signals Value Description 0 No effect. 1 Swaps the PhA and PhB signals.
0	ENABLE	R/W	0	Enable QEI Value Description 0 No effect. 1 Enables the quadrature encoder module.

Register 2: QEI Status (QEISTAT), offset 0x004

This register provides status about the operation of the QEI module.

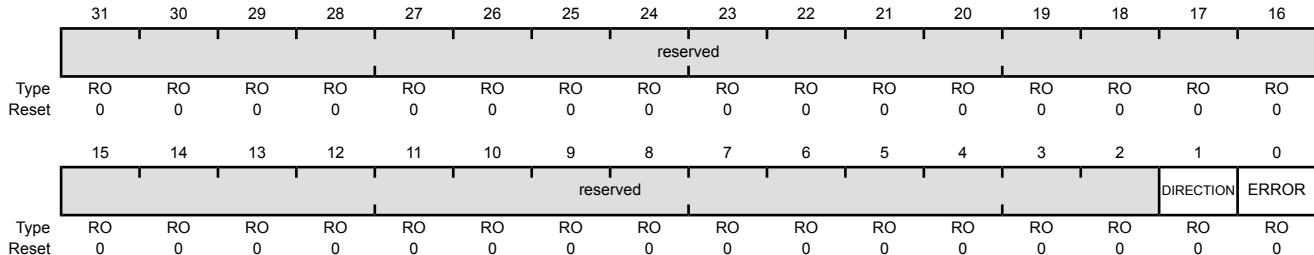
QEI Status (QEISTAT)

QEI0 base: 0x4002.C000

QEI1 base: 0x4002.D000

Offset 0x004

Type RO, reset 0x0000.0000



Bit/Field Name Type Reset Description

31:2 reserved RO 0x0000.0000 Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

1 DIRECTION RO 0 Direction of Rotation
Indicates the direction the encoder is rotating.

Value Description

- 0 The encoder is rotating forward.
- 1 The encoder is rotating in reverse.

0 ERROR RO 0 Error Detected

Value Description

- 0 No error.
- 1 An error was detected in the gray code sequence (that is, both signals changing at the same time).

Register 3: QEI Position (QEIPOS), offset 0x008

This register contains the current value of the position integrator. The value is updated by the status of the QEI phase inputs and can be set to a specific value by writing to it.

QEI Position (QEIPOS)

QEI0 base: 0x4002.C000

QEI1 base: 0x4002.D000

Offset 0x008

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	POSITION	R/W	0x0000.0000	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 in reverse, 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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	MAXPOS															
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
	MAXPOS															
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	MAXPOS	R/W	0x0000.0000	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. Because this value is loaded into the timer on 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 decimal clocks per timer period, this register should contain 1999 decimal.

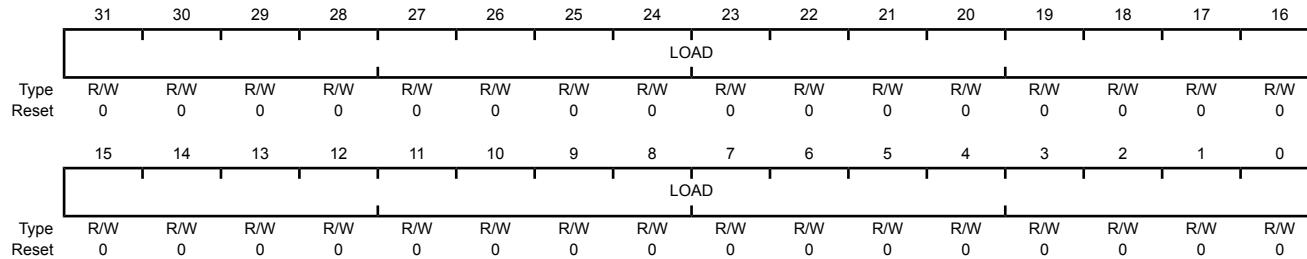
QEI Timer Load (QEILOAD)

QEI0 base: 0x4002.C000

QEI1 base: 0x4002.D000

Offset 0x010

Type R/W, reset 0x0000.0000



Bit/Field Name Type Reset Description

31:0 LOAD R/W 0x0000.0000 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 the VELEN bit in the **QEICTL** register is clear.

QEI Timer (QEITIME)

QEI0 base: 0x4002.C000

QEI1 base: 0x4002.D000

Offset 0x014

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TIME															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TIME															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	TIME	RO	0x0000.0000	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. Because this count 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 because there is a small window of time between the two reads, during which 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 the **VELEN** bit in the **QEICTL** register is clear.

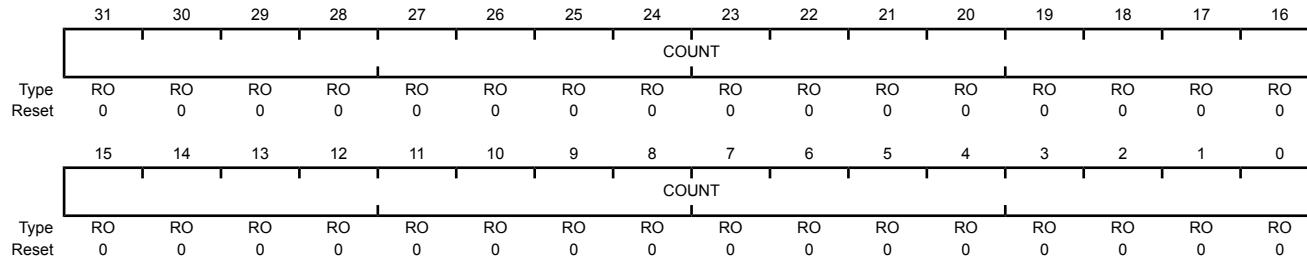
QEI Velocity Counter (QEICOUNT)

QEI0 base: 0x4002.C000

QEI1 base: 0x4002.D000

Offset 0x018

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	COUNT	RO	0x0000.0000	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 value corresponds to the number of velocity pulses counted in the previous velocity timer period. This register does not update when the VELEN bit in the **QEICTL** register is clear.

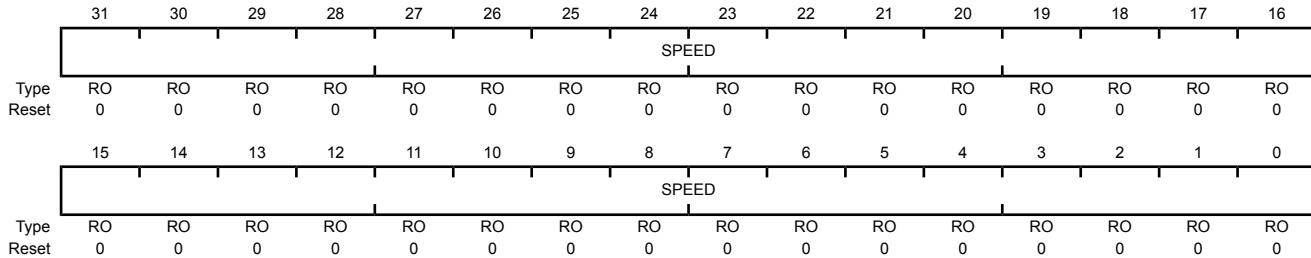
QEI Velocity (QEISPEED)

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	0x0000.0000	Velocity The measured speed of the quadrature encoder in pulses per period.

Register 9: QEI Interrupt Enable (QEINTEN), offset 0x020

This register contains enables for each of the QEI module interrupts. An interrupt is asserted to the interrupt controller if the corresponding bit in this register is set.

QEI Interrupt Enable (QEINTEN)

QEI0 base: 0x4002.C000
QEI1 base: 0x4002.D000
Offset 0x020
Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	INTERROR	R/W	0	Phase Error Interrupt Enable
	Value Description			
	1	An interrupt is sent to the interrupt controller when the INTERROR bit in the QEIRIS register is set.		
	0	The INTERROR interrupt is suppressed and not sent to the interrupt controller.		
2	INTDIR	R/W	0	Direction Change Interrupt Enable
	Value Description			
	1	An interrupt is sent to the interrupt controller when the INTDIR bit in the QEIRIS register is set.		
	0	The INTDIR interrupt is suppressed and not sent to the interrupt controller.		
1	INTTIMER	R/W	0	Timer Expires Interrupt Enable
	Value Description			
	1	An interrupt is sent to the interrupt controller when the INTTIMER bit in the QEIRIS register is set.		
	0	The INTTIMER interrupt is suppressed and not sent to the interrupt controller.		

Bit/Field	Name	Type	Reset	Description
0	INTINDEX	R/W	0	Index Pulse Detected Interrupt Enable
Value Description				
			1	An interrupt is sent to the interrupt controller when the INTINDEX bit in the QEIRIS register is set.
			0	The INTINDEX interrupt is suppressed and not sent to the interrupt controller.

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 (configured through the **QEINTEN** register). If a bit is set, the latched event has occurred; if a bit is clear, the event in question has not occurred.

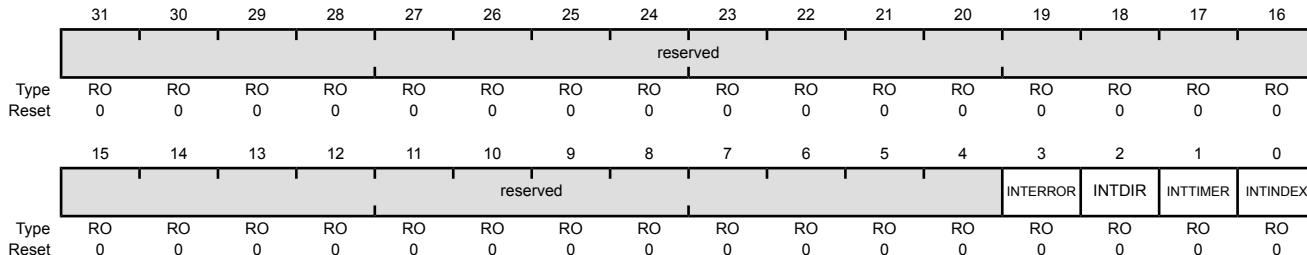
QEI Raw Interrupt Status (QEIRIS)

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	0x0000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	INTERROR	RO	0	<p>Phase Error Detected</p> <p>Value Description</p> <p>1 A phase error has been detected.</p> <p>0 An interrupt has not occurred.</p> <p>This bit is cleared by writing a 1 to the INTERROR bit in the QEISC register.</p>
2	INTDIR	RO	0	<p>Direction Change Detected</p> <p>Value Description</p> <p>1 The rotation direction has changed</p> <p>0 An interrupt has not occurred.</p> <p>This bit is cleared by writing a 1 to the INTDIR bit in the QEISC register.</p>
1	INTTIMER	RO	0	<p>Velocity Timer Expired</p> <p>Value Description</p> <p>1 The velocity timer has expired.</p> <p>0 An interrupt has not occurred.</p> <p>This bit is cleared by writing a 1 to the INTTIMER bit in the QEISC register.</p>

Bit/Field	Name	Type	Reset	Description
0	INTINDEX	RO	0	Index Pulse Asserted
				Value Description
				1 The index pulse has occurred.
				0 An interrupt has not occurred.
				This bit is cleared by writing a 1 to the INTINDEX bit in the QEIIISC register.

Register 11: QEI Interrupt Status and Clear (QEIIISC), offset 0x028

This register provides the current set of interrupt sources that are asserted to the controller. If a bit is set, the latched event has occurred and is enabled to generate an interrupt; if a bit is clear the event in question has not occurred or is not enabled to generate an interrupt. This register is R/W1C; writing a 1 to a bit position clears the bit and the corresponding interrupt reason.

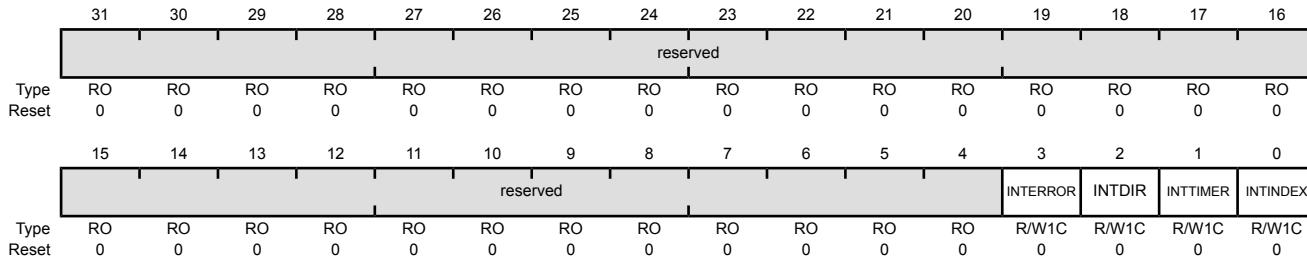
QEI Interrupt Status and Clear (QEIIISC)

QEI0 base: 0x4002.C000

QEI1 base: 0x4002.D000

Offset 0x028

Type R/W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	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	INTERROR	R/W1C	0	<p>Phase Error Interrupt</p> <p>Value Description</p> <p>1 The INTERROR bits in the QEIRIS register and the QEINTEN registers are set, providing an interrupt to the interrupt controller.</p> <p>0 No interrupt has occurred or the interrupt is masked.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INTERROR bit in the QEIRIS register.</p>
2	INTDIR	R/W1C	0	<p>Direction Change Interrupt</p> <p>Value Description</p> <p>1 The INTDIR bits in the QEIRIS register and the QEINTEN registers are set, providing an interrupt to the interrupt controller.</p> <p>0 No interrupt has occurred or the interrupt is masked.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INTDIR bit in the QEIRIS register.</p>
1	INTTIMER	R/W1C	0	<p>Velocity Timer Expired Interrupt</p> <p>Value Description</p> <p>1 The INTTIMER bits in the QEIRIS register and the QEINTEN registers are set, providing an interrupt to the interrupt controller.</p> <p>0 No interrupt has occurred or the interrupt is masked.</p> <p>This bit is cleared by writing a 1. Clearing this bit also clears the INTTIMER bit in the QEIRIS register.</p>

Bit/Field	Name	Type	Reset	Description
0	INTINDEX	R/W1C	0	Index Pulse Interrupt
Value Description				
			1	The INTINDEX bits in the QEIRIS register and the QEINTEN registers are set, providing an interrupt to the interrupt controller.
			0	No interrupt has occurred or the interrupt is masked.
This bit is cleared by writing a 1. Clearing this bit also clears the INTINDEX bit in the QEIRIS register.				

23 Pin Diagram

The LM3S9L97 microcontroller pin diagrams are 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 1190.

Figure 23-1. 100-Pin LQFP Package Pin Diagram

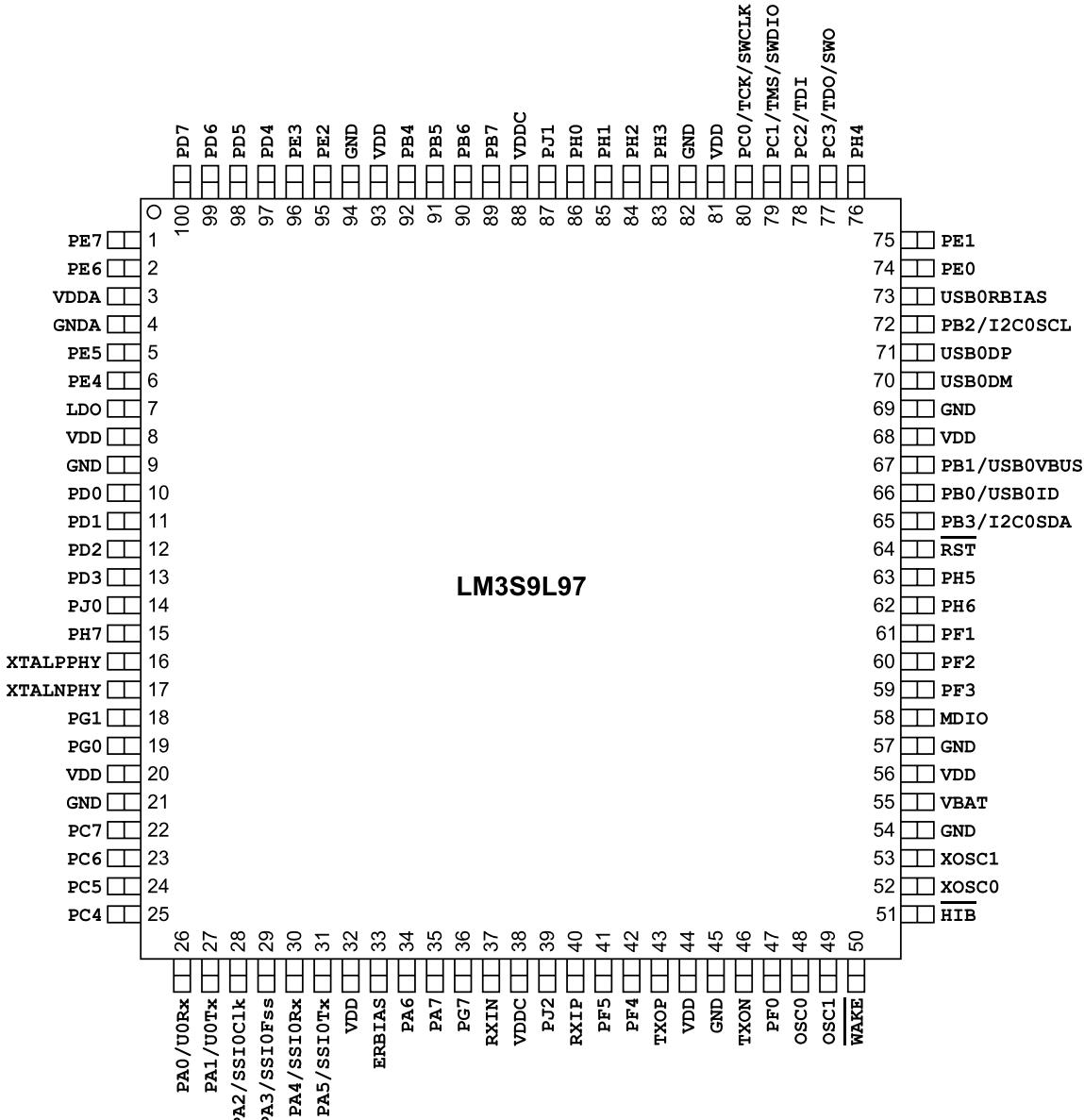
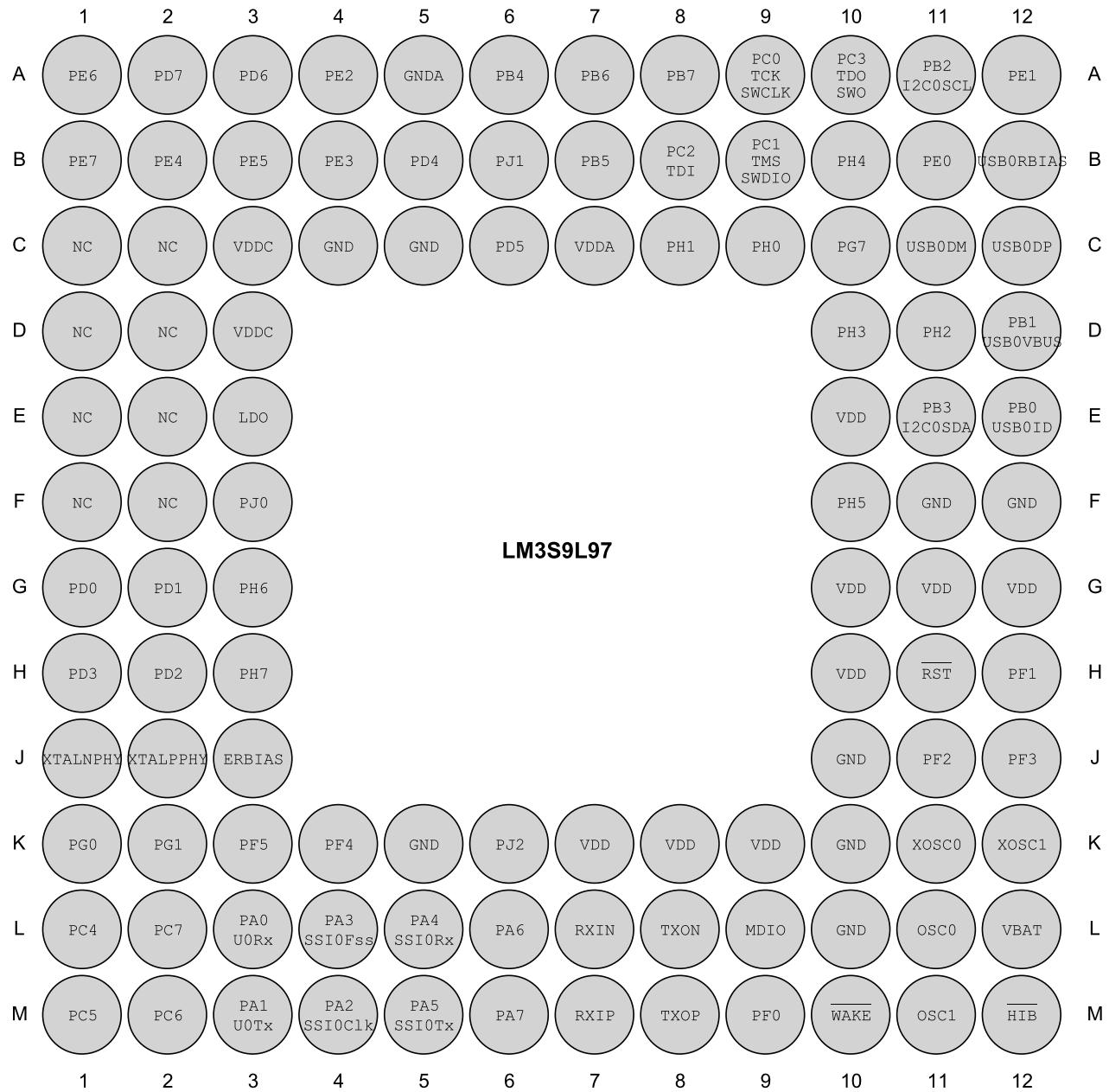


Figure 23-2. 108-Ball BGA Package Pin Diagram (Top View)

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. Use the **GPIOAMSEL** register (see page 463) to select analog mode. For a GPIO pin to be used for an alternate digital function, the corresponding bit in the **GPIOAFSEL** register (see page 447) must be set. Further pin muxing options are provided through the **PMCx** bit field in the **GPIOPCTL** register (see page 465), 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 the table below. A Power-On-Reset (**POR**) or asserting **RST** puts the pins back to their default state.

Table 24-1. GPIO Pins With Default Alternate Functions

GPIO Pin	Default State	GPIOAFSEL Bit	GPIOPCTL PMCx Bit Field
PA[1:0]	UART0	1	0x1
PA[5:2]	SSI0	1	0x1
PB[3:2]	I ² C0	1	0x1
PC[3:0]	JTAG/SWD	1	0x3

Table 24-2 on page 1163 shows the pin-to-signal-name mapping, including functional characteristics of the signals. Each possible alternate analog and digital function is listed for each pin.

Table 24-3 on page 1173 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 **PMCx** bit field in the **GPIOPCTL** register.

Table 24-4 on page 1182 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 1190 lists the GPIO pins and their analog and digital alternate functions. The **A_{IN}x** and **VREFA** analog signals are not 5-V tolerant and go through an isolation circuit before reaching their circuitry. These signals are configured by clearing the corresponding **DEN** bit in the **GPIO Digital Enable (GPIODEN)** register and setting the corresponding **AMSEL** bit in the **GPIO Analog Mode Select (GPIOAMSEL)** register. Other analog signals are 5-V tolerant and are connected directly to their circuitry (**C0-**, **C0+**, **C1-**, **C1+**, **USB0VBUS**, **USB0ID**). These signals are configured by clearing the **DEN** bit in the **GPIO Digital Enable (GPIODEN)** register. The digital signals are enabled by setting the appropriate bit in the **GPIO Alternate Function Select (GPIOAFSEL)** and **GPIODEN** registers and configuring the **PMCx** bit field in the **GPIO Port Control (GPIOPCTL)** register to the numeric encoding shown in the table below. Table entries that are shaded gray are the default values for the corresponding GPIO pin.

Table 24-6 on page 1193 lists the signals based on number of possible pin assignments. This table can be used to plan how to configure the pins for a particular functionality. Application Note AN01274 Configuring Stellaris® Microcontrollers with Pin Multiplexing provides an overview of the pin muxing implementation, an explanation of how a system designer defines a pin configuration, and examples of the pin configuration process.

Note: All digital inputs are Schmitt triggered.

24.1 100-Pin LQFP Package Pin Tables

Table 24-2. Signals by Pin Number

Pin Number	Pin Name	Pin Type	Buffer Type ^a	Description
1	PE7	I/O	TTL	GPIO port E bit 7.
	AIN0	I	Analog	Analog-to-digital converter input 0.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
	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	Analog-to-digital converter input 1.
	C1o	O	TTL	Analog comparator 1 output.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	U1CTS	I	TTL	UART module 1 Clear To Send modem status input signal.
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. VDDA pins must be connected to 3.3 V, regardless of system implementation.
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	Analog-to-digital converter input 2.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	I ² S0TXSD	I/O	TTL	I ² S module 0 transmit data.
6	PE4	I/O	TTL	GPIO port E bit 4.
	AIN3	I	Analog	Analog-to-digital converter input 3.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	Fault0	I	TTL	PWM Fault 0.
	I ² S0TXWS	I/O	TTL	I ² S module 0 transmit word select.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
7	LDO	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 μ 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 V _{DDC} 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.

Table 24-2. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type^a	Description
10	PDO	I/O	TTL	GPIO port D bit 0.
	AIN15	I	Analog	Analog-to-digital converter input 15.
	CAN0Rx	I	TTL	CAN module 0 receive.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	I2S0RXSCK	I/O	TTL	I ² S module 0 receive clock.
	IDX0	I	TTL	QEI module 0 index.
	PWM0	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
	U1CTS	I	TTL	UART module 1 Clear To Send modem status input signal.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
11	PD1	I/O	TTL	GPIO port D bit 1.
	AIN14	I	Analog	Analog-to-digital converter input 14.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	I2S0RXWS	I/O	TTL	I ² S module 0 receive word select.
	PWM1	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
	PhA0	I	TTL	QEI module 0 phase A.
	PhB1	I	TTL	QEI module 1 phase B.
	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
12	PD2	I/O	TTL	GPIO port D bit 2.
	AIN13	I	Analog	Analog-to-digital converter input 13.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	PWM2	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
13	PD3	I/O	TTL	GPIO port D bit 3.
	AIN12	I	Analog	Analog-to-digital converter input 12.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	PWM3	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
14	PJ0	I/O	TTL	GPIO port J bit 0.
	I2C1SCL	I/O	OD	I ² C module 1 clock.
	PWM0	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.

Table 24-2. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type^a	Description
15	PH7	I/O	TTL	GPIO port H bit 7.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
	SSI1Tx	O	TTL	SSI module 1 transmit.
16	XTALPPHY	I	Analog	Ethernet PHY XTALP 25-MHz oscillator crystal input.
17	XTALNPHY	O	Analog	Ethernet PHY XTALN 25-MHz oscillator crystal output.
18	PG1	I/O	TTL	GPIO port G bit 1.
	I2C1SDA	I/O	OD	I ² C module 1 data.
	PWM1	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
19	PG0	I/O	TTL	GPIO port G bit 0.
	I2C1SCL	I/O	OD	I ² C module 1 clock.
	PWM0	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
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.
	C1o	O	TTL	Analog comparator 1 output.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	PhB0	I	TTL	QEI module 0 phase B.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
23	PC6	I/O	TTL	GPIO port C bit 6.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	PhB0	I	TTL	QEI module 0 phase B.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.

Table 24-2. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type ^a	Description
24	PC5	I/O	TTL	GPIO port C bit 5.
	C0o	O	TTL	Analog comparator 0 output.
	C1+	I	Analog	Analog comparator 1 positive input.
	C1o	O	TTL	Analog comparator 1 output.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	Fault2	I	TTL	PWM Fault 2.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
25	PC4	I/O	TTL	GPIO port C bit 4.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	PhA0	I	TTL	QEI module 0 phase A.
26	PA0	I/O	TTL	GPIO port A bit 0.
	I2C1SCL	I/O	OD	I ² C module 1 clock.
	U0Rx	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
27	PA1	I/O	TTL	GPIO port A bit 1.
	I2C1SDA	I/O	OD	I ² C module 1 data.
	U0Tx	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
28	PA2	I/O	TTL	GPIO port A bit 2.
	I2S0RXSD	I/O	TTL	I ² S module 0 receive data.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	SSI0Clk	I/O	TTL	SSI module 0 clock.
29	PA3	I/O	TTL	GPIO port A bit 3.
	I2S0RXMCLK	I/O	TTL	I ² S module 0 receive master clock.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
	SSI0FSS	I/O	TTL	SSI module 0 frame.
30	PA4	I/O	TTL	GPIO port A bit 4.
	CAN0Rx	I	TTL	CAN module 0 receive.
	I2S0TXSCK	I/O	TTL	I ² S module 0 transmit clock.
	SSI0Rx	I	TTL	SSI module 0 receive.
31	PA5	I/O	TTL	GPIO port A bit 5.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	I2S0TXWS	I/O	TTL	I ² S module 0 transmit word select.
	SSI0Tx	O	TTL	SSI module 0 transmit.

Table 24-2. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type^a	Description
32	VDD	-	Power	Positive supply for I/O and some logic.
33	ERBIAS	O	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
34	PA6	I/O	TTL	GPIO port A bit 6.
	CAN0Rx	I	TTL	CAN module 0 receive.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	I2C1SCL	I/O	OD	I ² C module 1 clock.
	PWM0	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	U1CTS	I	TTL	UART module 1 Clear To Send modem status input signal.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
35	PA7	I/O	TTL	GPIO port A bit 7.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	I2C1SDA	I/O	OD	I ² C module 1 data.
	PWM1	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
36	PG7	I/O	TTL	GPIO port G bit 7.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	PhB1	I	TTL	QEI module 1 phase B.
37	RXIN	I	Analog	RXIN of the Ethernet PHY.
38	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.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	Fault0	I	TTL	PWM Fault 0.
40	RXIP	I	Analog	RXIP of the Ethernet PHY.
41	PF5	I/O	TTL	GPIO port F bit 5.
	C1o	O	TTL	Analog comparator 1 output.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	SSI1Tx	O	TTL	SSI module 1 transmit.
42	PF4	I/O	TTL	GPIO port F bit 4.
	C0o	O	TTL	Analog comparator 0 output.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	Fault0	I	TTL	PWM Fault 0.
	SSI1Rx	I	TTL	SSI module 1 receive.
43	TXOP	O	TTL	TXOP of the Ethernet PHY.
44	VDD	-	Power	Positive supply for I/O and some logic.

Table 24-2. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type^a	Description
45	GND	-	Power	Ground reference for logic and I/O pins.
46	TXON	O	TTL	TXON of the Ethernet PHY.
47	PF0	I/O	TTL	GPIO port F bit 0.
	CAN1Rx	I	TTL	CAN module 1 receive.
	I2S0TXSD	I/O	TTL	I ² S module 0 transmit data.
	PWM0	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
	PhB0	I	TTL	QEI module 0 phase B.
	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. Leave unconnected when using a single-ended clock source.
50	WAKE	I	TTL	An external input that brings the processor out of Hibernate mode when asserted.
51	HIB	O	OD	An open-drain output that indicates the processor is in Hibernate mode.
52	XOSC0	I	Analog	Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the HIBCTL register.
53	XOSC1	O	Analog	Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source.
54	GND	-	Power	Ground reference for logic and I/O pins.
55	VBAT	-	Power	Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply.
56	VDD	-	Power	Positive supply for I/O and some logic.
57	GND	-	Power	Ground reference for logic and I/O pins.
58	MDIO	I/O	OD	MDIO of the Ethernet PHY.
59	PF3	I/O	TTL	GPIO port F bit 3.
	LEDO	O	TTL	Ethernet LED 0.
	PWM3	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
	SSI1FSS	I/O	TTL	SSI module 1 frame.
60	PF2	I/O	TTL	GPIO port F bit 2.
	LED1	O	TTL	Ethernet LED 1.
	PWM2	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	SSI1Clk	I/O	TTL	SSI module 1 clock.

Table 24-2. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type ^a	Description
61	PF1	I/O	TTL	GPIO port F bit 1.
	CAN1Tx	O	TTL	CAN module 1 transmit.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	I2S0TXMCLK	I/O	TTL	I ² S module 0 transmit master clock.
	IDX1	I	TTL	QEI module 1 index.
	PWM1	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
	U1RTS	O	TTL	UART module 1 Request to Send modem output control line.
62	PH6	I/O	TTL	GPIO port H bit 6.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	SSI1Rx	I	TTL	SSI module 1 receive.
63	PH5	I/O	TTL	GPIO port H bit 5.
	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.
	Fault0	I	TTL	PWM Fault 0.
	Fault3	I	TTL	PWM Fault 3.
	I2C0SDA	I/O	OD	I ² C module 0 data.
	USB0PFLT	I	TTL	Optionally 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. This signal is controlled by PWM Generator 1.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	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 the A side of the cable and pulled up is the B side).
67	PB1	I/O	TTL	GPIO port B bit 1.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	PWM3	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	USB0VBUS	I/O	Analog	This signal is used during the session request 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).

Table 24-2. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type ^a	Description
72	PB2	I/O	TTL	GPIO port B bit 2.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	I2C0SCL	I/O	OD	I ² C module 0 clock.
	IDX0	I	TTL	QEI module 0 index.
	USB0EPEN	O	TTL	Optionally 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.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	SSI1Clk	I/O	TTL	SSI module 1 clock.
	USB0PFLT	I	TTL	Optionally 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.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	Fault0	I	TTL	PWM Fault 0.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
	SSI1FSS	I/O	TTL	SSI module 1 frame.
76	PH4	I/O	TTL	GPIO port H bit 4.
	SSI1Clk	I/O	TTL	SSI module 1 clock.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
77	PC3	I/O	TTL	GPIO port C bit 3.
	SWO	O	TTL	JTAG TDO and SWO.
	TDO	O	TTL	JTAG TDO and SWO.
78	PC2	I/O	TTL	GPIO port C bit 2.
	TDI	I	TTL	JTAG TDI.
79	PC1	I/O	TTL	GPIO port C bit 1.
	SWDIO	I/O	TTL	JTAG TMS and SWDIO.
	TMS	I	TTL	JTAG TMS and SWDIO.
80	PC0	I/O	TTL	GPIO port C bit 0.
	SWCLK	I	TTL	JTAG/SWD CLK.
	TCK	I	TTL	JTAG/SWD CLK.
81	VDD	-	Power	Positive supply for I/O and some logic.
82	GND	-	Power	Ground reference for logic and I/O pins.
83	PH3	I/O	TTL	GPIO port H bit 3.
	Fault0	I	TTL	PWM Fault 0.
	PhB0	I	TTL	QEI module 0 phase B.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.

Table 24-2. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type^a	Description
84	PH2	I/O	TTL	GPIO port H bit 2.
	C1o	O	TTL	Analog comparator 1 output.
	Fault3	I	TTL	PWM Fault 3.
	IDX1	I	TTL	QEI module 1 index.
85	PH1	I/O	TTL	GPIO port H bit 1.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	PWM3	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
86	PH0	I/O	TTL	GPIO port H bit 0.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	PWM2	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
87	PJ1	I/O	TTL	GPIO port J bit 1.
	I2C1SDA	I/O	OD	i ² C module 1 data.
	PWM1	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
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.
90	PB6	I/O	TTL	GPIO port B bit 6.
	C0+	I	Analog	Analog comparator 0 positive input.
	C0o	O	TTL	Analog comparator 0 output.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	Fault1	I	TTL	PWM Fault 1.
	I2S0TXSCK	I/O	TTL	I ² S module 0 transmit clock.
	IDX0	I	TTL	QEI module 0 index.
	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 _n signal is converted to 1023. The VREFA input is limited to the range specified in Table 26-25 on page 1244.

Table 24-2. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type ^a	Description
91	PB5	I/O	TTL	GPIO port B bit 5.
	AIN11	I	Analog	Analog-to-digital converter input 11.
	C0o	O	TTL	Analog comparator 0 output.
	C1-	I	Analog	Analog comparator 1 negative input.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
92	PB4	I/O	TTL	GPIO port B bit 4.
	AIN10	I	Analog	Analog-to-digital converter input 10.
	C0-	I	Analog	Analog comparator 0 negative input.
	CAN0Rx	I	TTL	CAN module 0 receive.
	IDX0	I	TTL	QEI module 0 index.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
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	Analog-to-digital converter input 9.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	PhA0	I	TTL	QEI module 0 phase A.
	PhB1	I	TTL	QEI module 1 phase B.
	SSI1Rx	I	TTL	SSI module 1 receive.
96	PE3	I/O	TTL	GPIO port E bit 3.
	AIN8	I	Analog	Analog-to-digital converter input 8.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	PhA1	I	TTL	QEI module 1 phase A.
	PhB0	I	TTL	QEI module 0 phase B.
	SSI1Tx	O	TTL	SSI module 1 transmit.
97	PD4	I/O	TTL	GPIO port D bit 4.
	AIN7	I	Analog	Analog-to-digital converter input 7.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	I2S0RXSD	I/O	TTL	I ² S module 0 receive data.
	U1RI	I	TTL	UART module 1 Ring Indicator modem status input signal.

Table 24-2. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type ^a	Description
98	PD5	I/O	TTL	GPIO port D bit 5.
	AIN6	I	Analog	Analog-to-digital converter input 6.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	I2S0RXMCLK	I/O	TTL	I ² S module 0 receive master clock.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
99	PD6	I/O	TTL	GPIO port D bit 6.
	AIN5	I	Analog	Analog-to-digital converter input 5.
	Fault0	I	TTL	PWM Fault 0.
	I2S0TXSCK	I/O	TTL	I ² S module 0 transmit clock.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
100	PD7	I/O	TTL	GPIO port D bit 7.
	AIN4	I	Analog	Analog-to-digital converter input 4.
	C0o	O	TTL	Analog comparator 0 output.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	I2S0TXWS	I/O	TTL	I ² S module 0 transmit word select.
	IDX0	I	TTL	QEI module 0 index.
	U1DTR	O	TTL	UART module 1 Data Terminal Ready modem status input signal.

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

Table 24-3. Signals by Signal Name

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
AIN0	1	PE7	I	Analog	Analog-to-digital converter input 0.
AIN1	2	PE6	I	Analog	Analog-to-digital converter input 1.
AIN2	5	PE5	I	Analog	Analog-to-digital converter input 2.
AIN3	6	PE4	I	Analog	Analog-to-digital converter input 3.
AIN4	100	PD7	I	Analog	Analog-to-digital converter input 4.
AIN5	99	PD6	I	Analog	Analog-to-digital converter input 5.
AIN6	98	PD5	I	Analog	Analog-to-digital converter input 6.
AIN7	97	PD4	I	Analog	Analog-to-digital converter input 7.
AIN8	96	PE3	I	Analog	Analog-to-digital converter input 8.
AIN9	95	PE2	I	Analog	Analog-to-digital converter input 9.
AIN10	92	PB4	I	Analog	Analog-to-digital converter input 10.
AIN11	91	PB5	I	Analog	Analog-to-digital converter input 11.
AIN12	13	PD3	I	Analog	Analog-to-digital converter input 12.
AIN13	12	PD2	I	Analog	Analog-to-digital converter input 13.
AIN14	11	PD1	I	Analog	Analog-to-digital converter input 14.
AIN15	10	PD0	I	Analog	Analog-to-digital converter input 15.
C0+	90	PB6	I	Analog	Analog comparator 0 positive input.

Table 24-3. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
C0-	92	PB4	I	Analog	Analog comparator 0 negative input.
C0o	24 42 90 91 100	PC5 (3) PF4 (2) PB6 (3) PB5 (1) PD7 (2)	O	TTL	Analog comparator 0 output.
C1+	24	PC5	I	Analog	Analog comparator 1 positive input.
C1-	91	PB5	I	Analog	Analog comparator 1 negative input.
C1o	2 22 24 41 84	PE6 (2) PC7 (7) PC5 (2) PF5 (2) PH2 (2)	O	TTL	Analog comparator 1 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.
CCP0	13 22 23 39 42 66 72 91 97	PD3 (4) PC7 (4) PC6 (6) PJ2 (9) PF4 (1) PB0 (1) PB2 (5) PB5 (4) PD4 (1)	I/O	TTL	Capture/Compare/PWM 0.
CCP1	24 25 34 67 90 96 100	PC5 (1) PC4 (9) PA6 (2) PB1 (4) PB6 (1) PE3 (1) PD7 (3)	I/O	TTL	Capture/Compare/PWM 1.
CCP2	6 11 25 41 67 75 91 95 98	PE4 (6) PD1 (10) PC4 (5) PF5 (1) PB1 (1) PE1 (4) PB5 (6) PE2 (5) PD5 (1)	I/O	TTL	Capture/Compare/PWM 2.

Table 24-3. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
CCP3	6 23 24 35 61 72 74 97	PE4 (1) PC6 (1) PC5 (5) PA7 (7) PF1 (10) PB2 (4) PE0 (3) PD4 (2)	I/O	TTL	Capture/Compare/PWM 3.
CCP4	22 25 35 95 98	PC7 (1) PC4 (6) PA7 (2) PE2 (1) PD5 (2)	I/O	TTL	Capture/Compare/PWM 4.
CCP5	5 12 25 36 90 91	PE5 (1) PD2 (4) PC4 (1) PG7 (8) PB6 (6) PB5 (2)	I/O	TTL	Capture/Compare/PWM 5.
CCP6	10 12 75 86 91	PD0 (6) PD2 (2) PE1 (5) PH0 (1) PB5 (3)	I/O	TTL	Capture/Compare/PWM 6.
CCP7	11 13 85 90 96	PD1 (6) PD3 (2) PH1 (1) PB6 (2) PE3 (5)	I/O	TTL	Capture/Compare/PWM 7.
ERBIAS	33	fixed	O	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
Fault0	6 39 42 65 75 83 99	PE4 (4) PJ2 (10) PF4 (4) PB3 (2) PE1 (3) PH3 (2) PD6 (1)	I	TTL	PWM Fault 0.
Fault1	90	PB6 (4)	I	TTL	PWM Fault 1.
Fault2	24 63	PC5 (4) PH5 (10)	I	TTL	PWM Fault 2.
Fault3	65 84	PB3 (4) PH2 (4)	I	TTL	PWM Fault 3.
GND	9 21 45 54 57 69 82 94	fixed	-	Power	Ground reference for logic and I/O pins.

Table 24-3. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
GNDA	4	fixed	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
HIB	51	fixed	O	OD	An open-drain output that indicates the processor is in Hibernate mode.
I2C0SCL	72	PB2 (1)	I/O	OD	I ² C module 0 clock.
I2C0SDA	65	PB3 (1)	I/O	OD	I ² C module 0 data.
I2C1SCL	14 19 26 34	PJ0 (11) PG0 (3) PA0 (8) PA6 (1)	I/O	OD	I ² C module 1 clock.
I2C1SDA	18 27 35 87	PG1 (3) PA1 (8) PA7 (1) PJ1 (11)	I/O	OD	I ² C module 1 data.
I2S0RXMCLK	29 98	PA3 (9) PD5 (8)	I/O	TTL	I ² S module 0 receive master clock.
I2S0RXSCK	10	PD0 (8)	I/O	TTL	I ² S module 0 receive clock.
I2S0RXSD	28 97	PA2 (9) PD4 (8)	I/O	TTL	I ² S module 0 receive data.
I2S0RXWS	11	PD1 (8)	I/O	TTL	I ² S module 0 receive word select.
I2S0TXMCLK	61	PF1 (8)	I/O	TTL	I ² S module 0 transmit master clock.
I2S0TXSCK	30 90 99	PA4 (9) PB6 (9) PD6 (8)	I/O	TTL	I ² S module 0 transmit clock.
I2S0TXSD	5 47	PE5 (9) PF0 (8)	I/O	TTL	I ² S module 0 transmit data.
I2S0TXWS	6 31 100	PE4 (9) PA5 (9) PD7 (8)	I/O	TTL	I ² S module 0 transmit word select.
IDX0	10 72 90 92 100	PD0 (3) PB2 (2) PB6 (5) PB4 (6) PD7 (1)	I	TTL	QEI module 0 index.
IDX1	61 84	PF1 (2) PH2 (1)	I	TTL	QEI module 1 index.
LDO	7	fixed	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 μ 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	Ethernet LED 0.
LED1	60	PF2 (1)	O	TTL	Ethernet LED 1.
MDIO	58	fixed	I/O	OD	MDIO of the Ethernet PHY.
NMI	89	PB7 (4)	I	TTL	Non-maskable interrupt.

Table 24-3. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
OSC0	48	fixed	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	49	fixed	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
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.
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.

Table 24-3. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
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	42	-	I/O	TTL	GPIO port F bit 4.
PF5	41	-	I/O	TTL	GPIO port F bit 5.
PG0	19	-	I/O	TTL	GPIO port G bit 0.
PG1	18	-	I/O	TTL	GPIO port G bit 1.
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.
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.
PhA0	11 25 95	PD1 (3) PC4 (2) PE2 (4)	I	TTL	QEI module 0 phase A.
PhA1	96	PE3 (3)	I	TTL	QEI module 1 phase A.
PhB0	22 23 47 83 96	PC7 (2) PC6 (2) 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.
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.
PWM0	10 14 19 34 47	PD0 (1) PJ0 (10) PG0 (2) PA6 (4) PF0 (3)	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
PWM1	11 18 35 61 87	PD1 (1) PG1 (2) PA7 (4) PF1 (3) PJ1 (10)	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.

Table 24-3. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
PWM2	12 60 66 86	PD2 (3) PF2 (4) PB0 (2) PH0 (2)	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
PWM3	13 59 67 85	PD3 (3) PF3 (4) PB1 (2) PH1 (2)	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
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. This signal is controlled by PWM Generator 2.
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. This signal is controlled by PWM Generator 2.
\overline{RST}	64	fixed	I	TTL	System reset input.
RXIN	37	fixed	I	Analog	RXIN of the Ethernet PHY.
RXIP	40	fixed	I	Analog	RXIP of the Ethernet PHY.
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	42 62 95	PF4 (9) PH6 (11) PE2 (2)	I	TTL	SSI module 1 receive.
SSI1Tx	15 41 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.

Table 24-3. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
TMS	79	PC1 (3)	I	TTL	JTAG TMS and SWDIO.
TXON	46	fixed	O	TTL	TXON of the Ethernet PHY.
TXOP	43	fixed	O	TTL	TXOP of the Ethernet PHY.
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	PE6 (9) PD0 (9) PA6 (9)	I	TTL	UART module 1 Clear To Send modem status input signal.
U1DCD	1 11 35	PE7 (9) PD1 (9) PA7 (9)	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
U1DSR	47	PF0 (9)	I	TTL	UART module 1 Data Set Ready modem output control line.
U1DTR	100	PD7 (9)	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
U1RI	97	PD4 (9)	I	TTL	UART module 1 Ring Indicator modem status input signal.
U1RTS	61	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. When in IrDA mode, this signal has IrDA modulation.
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. When in IrDA mode, this signal has IrDA modulation.
U2Rx	10 19 92 98	PD0 (4) PG0 (1) PB4 (4) PD5 (9)	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
U2Tx	6 11 18 99	PE4 (5) PD1 (4) PG1 (1) PD6 (9)	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
USB0DM	70	fixed	I/O	Analog	Bidirectional differential data pin (D- per USB specification).
USB0DP	71	fixed	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	Optionally used in Host mode to control an external power source to supply power to the USB bus.

Table 24-3. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
USB0ID	66	PB0	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 the A side of the cable and pulled up is the B side).
USB0PFLT	22 23 35 65 74 76 87	PC7 (6) PC6 (7) PA7 (8) PB3 (8) PE0 (9) PH4 (4) PJ1 (9)	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
USB0RBIAS	73	fixed	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.
USB0VBUS	67	PB1	I/O	Analog	This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing.
VBAT	55	fixed	-	Power	Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply.
VDD	8 20 32 44 56 68 81 93	fixed	-	Power	Positive supply for I/O and some logic.
VDDA	3	fixed	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be connected to 3.3 V, regardless of system implementation.
VDDC	38 88	fixed	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VREFA	90	PB6	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 _n signal is converted to 1023. The VREFA input is limited to the range specified in Table 26-25 on page 1244.
WAKE	50	fixed	I	TTL	An external input that brings the processor out of Hibernate mode when asserted.
XOSC0	52	fixed	I	Analog	Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the HIBCTL register.

Table 24-3. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
XOSC1	53	fixed	O	Analog	Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source.
XTALNPHY	17	fixed	O	Analog	Ethernet PHY XTALN 25-MHz oscillator crystal output.
XTALPPHY	16	fixed	I	Analog	Ethernet PHY XTALP 25-MHz oscillator crystal input.

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

Table 24-4. Signals by Function, Except for GPIO

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	Description
ADC	AIN0	1	I	Analog	Analog-to-digital converter input 0.
	AIN1	2	I	Analog	Analog-to-digital converter input 1.
	AIN2	5	I	Analog	Analog-to-digital converter input 2.
	AIN3	6	I	Analog	Analog-to-digital converter input 3.
	AIN4	100	I	Analog	Analog-to-digital converter input 4.
	AIN5	99	I	Analog	Analog-to-digital converter input 5.
	AIN6	98	I	Analog	Analog-to-digital converter input 6.
	AIN7	97	I	Analog	Analog-to-digital converter input 7.
	AIN8	96	I	Analog	Analog-to-digital converter input 8.
	AIN9	95	I	Analog	Analog-to-digital converter input 9.
	AIN10	92	I	Analog	Analog-to-digital converter input 10.
	AIN11	91	I	Analog	Analog-to-digital converter input 11.
	AIN12	13	I	Analog	Analog-to-digital converter input 12.
	AIN13	12	I	Analog	Analog-to-digital converter input 13.
	AIN14	11	I	Analog	Analog-to-digital converter input 14.
	AIN15	10	I	Analog	Analog-to-digital converter input 15.
	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 AINn signal is converted to 1023 . The VREFA input is limited to the range specified in Table 26-25 on page 1244.

Table 24-4. Signals by Function, Except for GPIO (*continued*)

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	Description
Analog Comparators	C0+	90	I	Analog	Analog comparator 0 positive input.
	C0-	92	I	Analog	Analog comparator 0 negative input.
	C0o	24 42 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 41 84	O	TTL	Analog comparator 1 output.
	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	ERBIAS	33	O	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
	LED0	59	O	TTL	Ethernet LED 0.
	LED1	60	O	TTL	Ethernet LED 1.
	MDIO	58	I/O	OD	MDIO of the Ethernet PHY.
	RXIN	37	I	Analog	RXIN of the Ethernet PHY.
	RXIP	40	I	Analog	RXIP of the Ethernet PHY.
	TXON	46	O	TTL	TXON of the Ethernet PHY.
	TXOP	43	O	TTL	TXOP of the Ethernet PHY.
	XTALNPHY	17	O	Analog	Ethernet PHY XTALN 25-MHz oscillator crystal output.
	XTALPPHY	16	I	Analog	Ethernet PHY XTALP 25-MHz oscillator crystal input.

Table 24-4. Signals by Function, Except for GPIO (*continued*)

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	Description
General-Purpose Timers	CCP0	13 22 23 39 42 66 72 91 97	I/O	TTL	Capture/Compare/PWM 0.
	CCP1	24 25 34 67 90 96 100	I/O	TTL	Capture/Compare/PWM 1.
	CCP2	6 11 25 41 67 75 91 95 98	I/O	TTL	Capture/Compare/PWM 2.
	CCP3	6 23 24 35 61 72 74 97	I/O	TTL	Capture/Compare/PWM 3.
	CCP4	22 25 35 95 98	I/O	TTL	Capture/Compare/PWM 4.
	CCP5	5 12 25 36 90 91	I/O	TTL	Capture/Compare/PWM 5.
	CCP6	10 12 75 86 91	I/O	TTL	Capture/Compare/PWM 6.
	CCP7	11 13 85 90 96	I/O	TTL	Capture/Compare/PWM 7.

Table 24-4. Signals by Function, Except for GPIO (continued)

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	Description
Hibernate	HIB	51	O	OD	An open-drain output that indicates the processor is in Hibernate mode.
	VBAT	55	-	Power	Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply.
	WAKE	50	I	TTL	An external input that brings the processor out of Hibernate mode when asserted.
	XOSC0	52	I	Analog	Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the HIBCTL register.
	XOSC1	53	O	Analog	Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source.
I ² C	I ² C0SCL	72	I/O	OD	I ² C module 0 clock.
	I ² C0SDA	65	I/O	OD	I ² C module 0 data.
	I ² C1SCL 14 19 26 34	14 19 26 34	I/O	OD	I ² C module 1 clock.
	I ² C1SDA 18 27 35 87	18 27 35 87	I/O	OD	I ² C module 1 data.
I ² S	I ² S0RXMCLK	29 98	I/O	TTL	I ² S module 0 receive master clock.
	I ² S0RXSCK	10	I/O	TTL	I ² S module 0 receive clock.
	I ² S0RXSD	28 97	I/O	TTL	I ² S module 0 receive data.
	I ² S0RXWS	11	I/O	TTL	I ² S module 0 receive word select.
	I ² S0TXMCLK	61	I/O	TTL	I ² S module 0 transmit master clock.
	I ² S0TXSCK	30 90 99	I/O	TTL	I ² S module 0 transmit clock.
	I ² S0TXSD	5 47	I/O	TTL	I ² S module 0 transmit data.
	I ² S0TXWS	6 31 100	I/O	TTL	I ² 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.

Table 24-4. Signals by Function, Except for GPIO (*continued*)

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	Description
PWM	Fault0	6 39 42 65 75 83 99	I	TTL	PWM Fault 0.
	Fault1	90	I	TTL	PWM Fault 1.
	Fault2	24 63	I	TTL	PWM Fault 2.
	Fault3	65 84	I	TTL	PWM Fault 3.
	PWM0	10 14 19 34 47	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
	PWM1	11 18 35 61 87	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
	PWM2	12 60 66 86	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
	PWM3	13 59 67 85	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
	PWM4	2 19 28 34 60 62 74 86	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	PWM5	1 15 18 29 35 59 75 85	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.

Table 24-4. Signals by Function, Except for GPIO (continued)

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	Description
Power	GND	9 21 45 54 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 μ 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. VDDA pins must be connected to 3.3 V, regardless of system implementation.
	VDDC	38 88	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
QEI	IDX0	10 72 90 92 100	I	TTL	QEI module 0 index.
	IDX1	61 84	I	TTL	QEI module 1 index.
	PhA0	11 25 95	I	TTL	QEI module 0 phase A.
	PhA1	96	I	TTL	QEI module 1 phase A.
	PhB0	22 23 47 83 96	I	TTL	QEI module 0 phase B.
	PhB1	11 36 95	I	TTL	QEI module 1 phase B.

Table 24-4. Signals by Function, Except for GPIO (*continued*)

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	Description
SSI	SSI0Clk	28	I/O	TTL	SSI module 0 clock.
	SSI0Fss	29	I/O	TTL	SSI module 0 frame.
	SSI0Rx	30	I	TTL	SSI module 0 receive.
	SSI0Tx	31	O	TTL	SSI module 0 transmit.
	SSI1Clk	60 74 76	I/O	TTL	SSI module 1 clock.
	SSI1Fss	59 63 75	I/O	TTL	SSI module 1 frame.
	SSI1Rx	42 62 95	I	TTL	SSI module 1 receive.
	SSI1Tx	15 41 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. Leave unconnected when using a single-ended clock source.
	RST	64	I	TTL	System reset input.

Table 24-4. Signals by Function, Except for GPIO (*continued*)

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	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	I	TTL	UART module 1 Clear To Send modem status input signal.
	U1DCD	1 11 35	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
	U1DSR	47	I	TTL	UART module 1 Data Set Ready modem output control line.
	U1DTR	100	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
	U1RI	97	I	TTL	UART module 1 Ring Indicator modem status input signal.
	U1RTS	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. When in IrDA mode, this signal has IrDA modulation.
	U1Tx	11 13 22 27 67 91	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	U2Rx	10 19 92 98	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
	U2Tx	6 11 18 99	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.

Table 24-4. Signals by Function, Except for GPIO (continued)

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	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	Optionally 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 the A side of the cable and pulled up is the B side).
	USB0PFLT	22 23 35 65 74 76 87	I	TTL	Optionally 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 request 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 has TTL-compatible voltage levels.

Table 24-5. GPIO Pins and Alternate Functions

IO	Pin	Analog Function	Digital Function (GPIO_PCTL_PMCx Bit Field Encoding) ^a										
			1	2	3	4	5	6	7	8	9	10	11
PA0	26	-	U0Rx	-	-	-	-	-	-	I2C1SCL	U1Rx	-	-
PA1	27	-	U0Tx	-	-	-	-	-	-	I2C1SDA	U1Tx	-	-
PA2	28	-	SSI0Clk	-	-	PWM4	-	-	-	-	I2S0RXSD	-	-
PA3	29	-	SSI0Fss	-	-	PWM5	-	-	-	-	I2S0RXMCLK	-	-
PA4	30	-	SSI0Rx	-	-	-	CAN0Rx	-	-	-	I2S0TXSCK	-	-
PA5	31	-	SSI0Tx	-	-	-	CAN0Tx	-	-	-	I2S0TXWS	-	-
PA6	34	-	I2C1SCL	CCP1	-	PWM0	PWM4	CAN0Rx	-	USB0EPEN	U1CTS	-	-
PA7	35	-	I2C1SDA	CCP4	-	PWM1	PWM5	CAN0Tx	CCP3	USB0PFLT	U1DCD	-	-
PB0	66	USB0ID	CCP0	PWM2	-	-	U1Rx	-	-	-	-	-	-
PB1	67	USB0VBUS	CCP2	PWM3	-	CCP1	U1Tx	-	-	-	-	-	-
PB2	72	-	I2C0SCL	IDX0	-	CCP3	CCP0	-	-	USB0EPEN	-	-	-
PB3	65	-	I2C0SDA	Fault0	-	Fault3	-	-	-	USB0PFLT	-	-	-
PB4	92	AIN10 C0-	-	-	-	U2Rx	CAN0Rx	IDX0	U1Rx	-	-	-	-
PB5	91	AIN11 C1-	C0o	CCP5	CCP6	CCP0	CAN0Tx	CCP2	U1Tx	-	-	-	-

Table 24-5. GPIO Pins and Alternate Functions (continued)

IO	Pin	Analog Function	Digital Function (GPIO PCTL PMCx Bit Field Encoding) ^a										
			1	2	3	4	5	6	7	8	9	10	11
PB6	90	VREFA C0+	CCP1	CCP7	C0o	Fault1	IDX0	CCP5	-	-	I2S0TXSCK	-	-
PB7	89	-	-	-	-	NMI	-	-	-	-	-	-	-
PC0	80	-	-	-	TCK SWCLK	-	-	-	-	-	-	-	-
PC1	79	-	-	-	TMS SWDIO	-	-	-	-	-	-	-	-
PC2	78	-	-	-	TDI	-	-	-	-	-	-	-	-
PC3	77	-	-	-	TDO SWO	-	-	-	-	-	-	-	-
PC4	25	-	CCP5	PhA0	-	-	CCP2	CCP4	-	-	CCP1	-	-
PC5	24	C1+	CCP1	C1o	C0o	Fault2	CCP3	USB0EPEN	-	-	-	-	-
PC6	23	-	CCP3	PhB0	-	-	U1Rx	CCP0	USB0PFLT	-	-	-	-
PC7	22	-	CCP4	PhB0	-	CCP0	U1Tx	USB0PFLT	C1o	-	-	-	-
PD0	10	AIN15	PWM0	CAN0Rx	IDX0	U2Rx	U1Rx	CCP6	-	I2S0RXSCK	U1CTS	-	-
PD1	11	AIN14	PWM1	CAN0Tx	PhA0	U2Tx	U1Tx	CCP7	-	I2S0RXWS	U1DCD	CCP2	PhB1
PD2	12	AIN13	U1Rx	CCP6	PWM2	CCP5	-	-	-	-	-	-	-
PD3	13	AIN12	U1Tx	CCP7	PWM3	CCP0	-	-	-	-	-	-	-
PD4	97	AIN7	CCP0	CCP3	-	-	-	-	-	I2S0RXSD	U1RI	-	-
PD5	98	AIN6	CCP2	CCP4	-	-	-	-	-	I2S0RXMCLK	U2Rx	-	-
PD6	99	AIN5	Fault0	-	-	-	-	-	-	I2S0TXSCK	U2Tx	-	-
PD7	100	AIN4	IDX0	C0o	CCP1	-	-	-	-	I2S0TXWS	U1DTR	-	-
PE0	74	-	PWM4	SSI1Clk	CCP3	-	-	-	-	-	USB0PFLT	-	-
PE1	75	-	PWM5	SSI1Fss	Fault0	CCP2	CCP6	-	-	-	-	-	-
PE2	95	AIN9	CCP4	SSI1Rx	PhB1	PhA0	CCP2	-	-	-	-	-	-
PE3	96	AIN8	CCP1	SSI1Tx	PhA1	PhB0	CCP7	-	-	-	-	-	-
PE4	6	AIN3	CCP3	-	-	Fault0	U2Tx	CCP2	-	-	I2S0TXWS	-	-
PE5	5	AIN2	CCP5	-	-	-	-	-	-	-	I2S0TXSD	-	-
PE6	2	AIN1	PWM4	C1o	-	-	-	-	-	-	U1CTS	-	-
PE7	1	AIN0	PWM5	-	-	-	-	-	-	-	U1DCD	-	-
PF0	47	-	CAN1Rx	PhB0	PWM0	-	-	-	-	I2S0TXSD	U1DSR	-	-
PF1	61	-	CAN1Tx	IDX1	PWM1	-	-	-	-	I2S0TXMCLK	U1RTS	CCP3	-
PF2	60	-	LED1	PWM4	-	PWM2	-	-	-	-	SSI1Clk	-	-
PF3	59	-	LED0	PWM5	-	PWM3	-	-	-	-	SSI1Fss	-	-
PF4	42	-	CCP0	C0o	-	Fault0	-	-	-	-	SSI1Rx	-	-
PF5	41	-	CCP2	C1o	-	-	-	-	-	-	SSI1Tx	-	-
PG0	19	-	U2Rx	PWM0	I2C1SCL	PWM4	-	-	USBOEPEN	-	-	-	-
PG1	18	-	U2Tx	PWM1	I2C1SDA	PWM5	-	-	-	-	-	-	-
PG7	36	-	PhB1	-	-	-	-	-	-	CCP5	-	-	-
PH0	86	-	CCP6	PWM2	-	-	-	-	-	-	PWM4	-	-
PH1	85	-	CCP7	PWM3	-	-	-	-	-	-	PWM5	-	-
PH2	84	-	IDX1	C1o	-	Fault3	-	-	-	-	-	-	-

Table 24-5. GPIO Pins and Alternate Functions (continued)

IO	Pin	Analog Function	Digital Function (GPIO PCTL PMCx Bit Field Encoding) ^a										
			1	2	3	4	5	6	7	8	9	10	11
PH3	83	-	PhB0	Fault0	-	USB0OPEN	-	-	-	-	-	-	-
PH4	76	-	-	-	-	USB0PFLT	-	-	-	-	-	-	SSI1Clk
PH5	63	-	-	-	-	-	-	-	-	-	-	Fault2	SSI1Fss
PH6	62	-	-	-	-	-	-	-	-	-	-	PWM4	SSI1Rx
PH7	15	-	-	-	-	-	-	-	-	-	-	PWM5	SSI1Tx
PJ0	14	-	-	-	-	-	-	-	-	-	-	PWM0	I2C1SCL
PJ1	87	-	-	-	-	-	-	-	-	-	USB0PFLT	PWM1	I2C1SDA
PJ2	39	-	-	-	-	-	-	-	-	-	CCP0	Fault0	-

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin.

Table 24-6. Possible Pin Assignments for Alternate Functions

# of Possible Assignments	Alternate Function	GPIO Function
one	AIN0	PE7
	AIN1	PE6
	AIN10	PB4
	AIN11	PB5
	AIN12	PD3
	AIN13	PD2
	AIN14	PD1
	AIN15	PD0
	AIN2	PE5
	AIN3	PE4
	AIN4	PD7
	AIN5	PD6
	AIN6	PD5
	AIN7	PD4
	AIN8	PE3
	AIN9	PE2
	C0+	PB6
	C0-	PB4
	C1+	PC5
	C1-	PB5
	CAN1Rx	PF0
	CAN1Tx	PF1
	Fault1	PB6
	I2C0SCL	PB2
	I2C0SDA	PB3
	I2S0RXSCK	PD0
	I2S0RXWS	PD1
	I2S0TXMCLK	PF1
	LED0	PF3
	LED1	PF2
	NMI	PB7
	PhA1	PE3
	SSI0Clk	PA2
	SSI0FSS	PA3
	SSI0Rx	PA4
	SSI0Tx	PA5
	SWCLK	PC0
	SWDIO	PC1
	SWO	PC3
	TCK	PC0
	TDI	PC2

Table 24-6. Possible Pin Assignments for Alternate Functions (continued)

# of Possible Assignments	Alternate Function	GPIO Function
two	TDO	PC3
	TMS	PC1
	U0Rx	PA0
	U0Tx	PA1
	U1DSR	PF0
	U1DTR	PD7
	U1RI	PD4
	U1RTS	PF1
	USB0ID	PB0
	USB0VBUS	PB1
three	VREFA	PB6
	Fault2	PC5 PH5
	Fault3	PB3 PH2
	I2S0RXMCLK	PA3 PD5
	I2S0RXSD	PA2 PD4
	I2S0TXSD	PE5 PF0
four	IDX1	PF1 PH2
	I2S0TXSCK	PA4 PB6 PD6
	I2S0TXWS	PA5 PD7 PE4
	PhA0	PC4 PD1 PE2
	PhB1	PD1 PE2 PG7
	SSI1Clk	PE0 PF2 PH4
	SSI1FSS	PE1 PF3 PH5
	SSI1Rx	PE2 PF4 PH6
	SSI1Tx	PE3 PF5 PH7
	U1CTS	PA6 PD0 PE6
four	U1DCD	PA7 PD1 PE7
	CAN0Rx	PA4 PA6 PB4 PD0
	CAN0Tx	PA5 PA7 PB5 PD1
	I2C1SCL	PA0 PA6 PG0 PJ0
	I2C1SDA	PA1 PA7 PG1 PJ1
	PWM2	PB0 PD2 PF2 PH0
	PWM3	PB1 PD3 PF3 PH1
	U2Rx	PB4 PD0 PD5 PG0
	U2Tx	PD1 PD6 PE4 PG1

Table 24-6. Possible Pin Assignments for Alternate Functions (continued)

# of Possible Assignments	Alternate Function	GPIO Function
five	C0o	PB5 PB6 PC5 PD7 PF4
	C1o	PC5 PC7 PE6 PF5 PH2
	CCP4	PA7 PC4 PC7 PD5 PE2
	CCP6	PB5 PD0 PD2 PE1 PH0
	CCP7	PB6 PD1 PD3 PE3 PH1
	IDX0	PB2 PB4 PB6 PD0 PD7
	PWM0	PA6 PD0 PF0 PG0 PJ0
	PWM1	PA7 PD1 PF1 PG1 PJ1
	PhB0	PC6 PC7 PE3 PF0 PH3
	USB0EPEN	PA6 PB2 PC5 PG0 PH3
six	CCP5	PB5 PB6 PC4 PD2 PE5 PG7
	U1Rx	PA0 PB0 PB4 PC6 PD0 PD2
	U1Tx	PA1 PB1 PB5 PC7 PD1 PD3
seven	CCP1	PA6 PB1 PB6 PC4 PC5 PD7 PE3
	Fault0	PB3 PD6 PE1 PE4 PF4 PH3 PJ2
	USB0PFLT	PA7 PB3 PC6 PC7 PE0 PH4 PJ1
eight	CCP3	PA7 PB2 PC5 PC6 PD4 PE0 PE4 PF1
	PWM4	PA2 PA6 PE0 PE6 PF2 PG0 PH0 PH6
	PWM5	PA3 PA7 PE1 PE7 PF3 PG1 PH1 PH7
nine	CCP0	PB0 PB2 PB5 PC6 PC7 PD3 PD4 PF4 PJ2
	CCP2	PB1 PB5 PC4 PD1 PD5 PE1 PE2 PE4 PF5

24.2 108-Pin BGA Package Pin Tables

Table 24-7. Signals by Pin Number

Pin Number	Pin Name	Pin Type	Buffer Type ^a	Description
A1	PE6	I/O	TTL	GPIO port E bit 6.
	AIN1	I	Analog	Analog-to-digital converter input 1.
	C1o	O	TTL	Analog comparator 1 output.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	U1CTS	I	TTL	UART module 1 Clear To Send modem status input signal.
A2	PD7	I/O	TTL	GPIO port D bit 7.
	AIN4	I	Analog	Analog-to-digital converter input 4.
	C0o	O	TTL	Analog comparator 0 output.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	I2S0TXWS	I/O	TTL	I ² S module 0 transmit word select.
	IDX0	I	TTL	QEI module 0 index.
	U1DTR	O	TTL	UART module 1 Data Terminal Ready modem status input signal.

Table 24-7. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type ^a	Description
A3	PD6	I/O	TTL	GPIO port D bit 6.
	AIN5	I	Analog	Analog-to-digital converter input 5.
	Fault0	I	TTL	PWM Fault 0.
	I ² S0TXSCK	I/O	TTL	I ² S module 0 transmit clock.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
A4	PE2	I/O	TTL	GPIO port E bit 2.
	AIN9	I	Analog	Analog-to-digital converter input 9.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	PhA0	I	TTL	QEI module 0 phase A.
	PhB1	I	TTL	QEI module 1 phase B.
	SSI1Rx	I	TTL	SSI module 1 receive.
A5	GNDA	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
A6	PB4	I/O	TTL	GPIO port B bit 4.
	AIN10	I	Analog	Analog-to-digital converter input 10.
	C0-	I	Analog	Analog comparator 0 negative input.
	CAN0Rx	I	TTL	CAN module 0 receive.
	IDX0	I	TTL	QEI module 0 index.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
A7	PB6	I/O	TTL	GPIO port B bit 6.
	C0+	I	Analog	Analog comparator 0 positive input.
	C0o	O	TTL	Analog comparator 0 output.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	Fault1	I	TTL	PWM Fault 1.
	I ² S0TXSCK	I/O	TTL	I ² S module 0 transmit clock.
	IDX0	I	TTL	QEI module 0 index.
A8	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 AINn signal is converted to 1023. The VREFA input is limited to the range specified in Table 26-25 on page 1244.
	PB7	I/O	TTL	GPIO port B bit 7.
	NMI	I	TTL	Non-maskable interrupt.

Table 24-7. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type^a	Description
A9	PC0	I/O	TTL	GPIO port C bit 0.
	SWCLK	I	TTL	JTAG/SWD CLK.
	TCK	I	TTL	JTAG/SWD CLK.
A10	PC3	I/O	TTL	GPIO port C bit 3.
	SWO	O	TTL	JTAG TDO and SWO.
	TDO	O	TTL	JTAG TDO and SWO.
A11	PB2	I/O	TTL	GPIO port B bit 2.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	I2C0SCL	I/O	OD	I ² C module 0 clock.
	IDX0	I	TTL	QEI module 0 index.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
A12	PE1	I/O	TTL	GPIO port E bit 1.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	Fault0	I	TTL	PWM Fault 0.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
	SSI1FSS	I/O	TTL	SSI module 1 frame.
B1	PE7	I/O	TTL	GPIO port E bit 7.
	AIN0	I	Analog	Analog-to-digital converter input 0.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
B2	PE4	I/O	TTL	GPIO port E bit 4.
	AIN3	I	Analog	Analog-to-digital converter input 3.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	Fault0	I	TTL	PWM Fault 0.
	I2S0TXWS	I/O	TTL	I ² S module 0 transmit word select.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
B3	PE5	I/O	TTL	GPIO port E bit 5.
	AIN2	I	Analog	Analog-to-digital converter input 2.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	I2S0TXSD	I/O	TTL	I ² S module 0 transmit data.
B4	PE3	I/O	TTL	GPIO port E bit 3.
	AIN8	I	Analog	Analog-to-digital converter input 8.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	PhA1	I	TTL	QEI module 1 phase A.
	PhB0	I	TTL	QEI module 0 phase B.
	SSI1Tx	O	TTL	SSI module 1 transmit.

Table 24-7. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type ^a	Description
B5	PD4	I/O	TTL	GPIO port D bit 4.
	AIN7	I	Analog	Analog-to-digital converter input 7.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	I ² S0RXSD	I/O	TTL	I ² S module 0 receive data.
	U1RI	I	TTL	UART module 1 Ring Indicator modem status input signal.
B6	PJ1	I/O	TTL	GPIO port J bit 1.
	I ² C1SDA	I/O	OD	I ² C module 1 data.
	PWM1	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
B7	PB5	I/O	TTL	GPIO port B bit 5.
	AIN11	I	Analog	Analog-to-digital converter input 11.
	C0o	O	TTL	Analog comparator 0 output.
	C1-	I	Analog	Analog comparator 1 negative input.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
B8	PC2	I/O	TTL	GPIO port C bit 2.
	TDI	I	TTL	JTAG TDI.
B9	PC1	I/O	TTL	GPIO port C bit 1.
	SWDIO	I/O	TTL	JTAG TMS and SWDIO.
	TMS	I	TTL	JTAG TMS and SWDIO.
B10	PH4	I/O	TTL	GPIO port H bit 4.
	SSI1Clk	I/O	TTL	SSI module 1 clock.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
B11	PE0	I/O	TTL	GPIO port E bit 0.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	SSI1Clk	I/O	TTL	SSI module 1 clock.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
B12	USB0RBIAS	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.
C1	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
C2	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
C3	VDDC	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.

Table 24-7. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type^a	Description
C4	GND	-	Power	Ground reference for logic and I/O pins.
C5	GND	-	Power	Ground reference for logic and I/O pins.
C6	PD5	I/O	TTL	GPIO port D bit 5.
	AIN6	I	Analog	Analog-to-digital converter input 6.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	I ² S0RXMCLK	I/O	TTL	I ² S module 0 receive master clock.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
C7	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. VDDA pins must be connected to 3.3 V, regardless of system implementation.
C8	PH1	I/O	TTL	GPIO port H bit 1.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	PWM3	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
C9	PH0	I/O	TTL	GPIO port H bit 0.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	PWM2	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
C10	PG7	I/O	TTL	GPIO port G bit 7.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	PhB1	I	TTL	QEI module 1 phase B.
C11	USB0DM	I/O	Analog	Bidirectional differential data pin (D- per USB specification).
C12	USB0DP	I/O	Analog	Bidirectional differential data pin (D+ per USB specification).
D1	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
D2	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
D3	VDDC	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
D10	PH3	I/O	TTL	GPIO port H bit 3.
	Fault0	I	TTL	PWM Fault 0.
	PhB0	I	TTL	QEI module 0 phase B.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
D11	PH2	I/O	TTL	GPIO port H bit 2.
	C1o	O	TTL	Analog comparator 1 output.
	Fault3	I	TTL	PWM Fault 3.
	IDX1	I	TTL	QEI module 1 index.

Table 24-7. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type ^a	Description
D12	PB1	I/O	TTL	GPIO port B bit 1.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	PWM3	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	USB0VBUS	I/O	Analog	This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing.
E1	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
E2	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
E3	LDO	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 μ 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).
E10	VDD	-	Power	Positive supply for I/O and some logic.
E11	PB3	I/O	TTL	GPIO port B bit 3.
	Fault0	I	TTL	PWM Fault 0.
	Fault3	I	TTL	PWM Fault 3.
	I2C0SDA	I/O	OD	I ² C module 0 data.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
E12	PB0	I/O	TTL	GPIO port B bit 0.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	PWM2	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	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 the A side of the cable and pulled up is the B side).
F1	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
F2	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
F3	PJ0	I/O	TTL	GPIO port J bit 0.
	I2C1SCL	I/O	OD	I ² C module 1 clock.
	PWM0	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
F10	PH5	I/O	TTL	GPIO port H bit 5.
	Fault2	I	TTL	PWM Fault 2.
	SSI1Fss	I/O	TTL	SSI module 1 frame.
F11	GND	-	Power	Ground reference for logic and I/O pins.
F12	GND	-	Power	Ground reference for logic and I/O pins.

Table 24-7. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type^a	Description
G1	PDO	I/O	TTL	GPIO port D bit 0.
	AIN15	I	Analog	Analog-to-digital converter input 15.
	CAN0Rx	I	TTL	CAN module 0 receive.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	I2S0RXSCK	I/O	TTL	I ² S module 0 receive clock.
	IDX0	I	TTL	QEI module 0 index.
	PWM0	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
	U1CTS	I	TTL	UART module 1 Clear To Send modem status input signal.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
G2	PD1	I/O	TTL	GPIO port D bit 1.
	AIN14	I	Analog	Analog-to-digital converter input 14.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	I2S0RXWS	I/O	TTL	I ² S module 0 receive word select.
	PWM1	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
	PhA0	I	TTL	QEI module 0 phase A.
	PhB1	I	TTL	QEI module 1 phase B.
	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
G3	PH6	I/O	TTL	GPIO port H bit 6.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	SSI1Rx	I	TTL	SSI module 1 receive.
G10	VDD	-	Power	Positive supply for I/O and some logic.
G11	VDD	-	Power	Positive supply for I/O and some logic.
G12	VDD	-	Power	Positive supply for I/O and some logic.
H1	PD3	I/O	TTL	GPIO port D bit 3.
	AIN12	I	Analog	Analog-to-digital converter input 12.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP7	I/O	TTL	Capture/Compare/PWM 7.
	PWM3	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.

Table 24-7. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type ^a	Description
H2	PD2	I/O	TTL	GPIO port D bit 2.
	AIN13	I	Analog	Analog-to-digital converter input 13.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	CCP6	I/O	TTL	Capture/Compare/PWM 6.
	PWM2	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
H3	PH7	I/O	TTL	GPIO port H bit 7.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
	SSI1Tx	O	TTL	SSI module 1 transmit.
H10	VDD	-	Power	Positive supply for I/O and some logic.
H11	RST	I	TTL	System reset input.
H12	PF1	I/O	TTL	GPIO port F bit 1.
	CAN1Tx	O	TTL	CAN module 1 transmit.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	I2S0TXMCLK	I/O	TTL	I ² S module 0 transmit master clock.
	IDX1	I	TTL	QEI module 1 index.
	PWM1	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
	U1RTS	O	TTL	UART module 1 Request to Send modem output control line.
J1	XTALNPHY	O	Analog	Ethernet PHY XTALN 25-MHz oscillator crystal output.
J2	XTALPPHY	I	Analog	Ethernet PHY XTALP 25-MHz oscillator crystal input.
J3	ERBIAS	O	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
J10	GND	-	Power	Ground reference for logic and I/O pins.
J11	PF2	I/O	TTL	GPIO port F bit 2.
	LED1	O	TTL	Ethernet LED 1.
	PWM2	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	SSI1Clk	I/O	TTL	SSI module 1 clock.
J12	PF3	I/O	TTL	GPIO port F bit 3.
	LEDO	O	TTL	Ethernet LED 0.
	PWM3	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
	SSI1FSS	I/O	TTL	SSI module 1 frame.
K1	PG0	I/O	TTL	GPIO port G bit 0.
	I2C1SCL	I/O	OD	I ² C module 1 clock.
	PWM0	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.

Table 24-7. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type ^a	Description
K2	PG1	I/O	TTL	GPIO port G bit 1.
	I2C1SDA	I/O	OD	I ² C module 1 data.
	PWM1	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
	U2Tx	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
K3	PF5	I/O	TTL	GPIO port F bit 5.
	C1o	O	TTL	Analog comparator 1 output.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	SSI1Tx	O	TTL	SSI module 1 transmit.
K4	PF4	I/O	TTL	GPIO port F bit 4.
	C0o	O	TTL	Analog comparator 0 output.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	Fault0	I	TTL	PWM Fault 0.
	SSI1Rx	I	TTL	SSI module 1 receive.
K5	GND	-	Power	Ground reference for logic and I/O pins.
K6	PJ2	I/O	TTL	GPIO port J bit 2.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	Fault0	I	TTL	PWM Fault 0.
K7	VDD	-	Power	Positive supply for I/O and some logic.
K8	VDD	-	Power	Positive supply for I/O and some logic.
K9	VDD	-	Power	Positive supply for I/O and some logic.
K10	GND	-	Power	Ground reference for logic and I/O pins.
K11	XOSC0	I	Analog	Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the HIBCTL register.
K12	XOSC1	O	Analog	Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source.
L1	PC4	I/O	TTL	GPIO port C bit 4.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	PhA0	I	TTL	QEI module 0 phase A.

Table 24-7. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type ^a	Description
L2	PC7	I/O	TTL	GPIO port C bit 7.
	C1o	O	TTL	Analog comparator 1 output.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	PhB0	I	TTL	QEI module 0 phase B.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
L3	PA0	I/O	TTL	GPIO port A bit 0.
	I2C1SCL	I/O	OD	I ² C module 1 clock.
	U0Rx	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
L4	PA3	I/O	TTL	GPIO port A bit 3.
	I2S0RXMCLK	I/O	TTL	I ² S module 0 receive master clock.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
	SSI0FSS	I/O	TTL	SSI module 0 frame.
L5	PA4	I/O	TTL	GPIO port A bit 4.
	CAN0Rx	I	TTL	CAN module 0 receive.
	I2S0TXSCK	I/O	TTL	I ² S module 0 transmit clock.
	SSI0Rx	I	TTL	SSI module 0 receive.
L6	PA6	I/O	TTL	GPIO port A bit 6.
	CAN0Rx	I	TTL	CAN module 0 receive.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	I2C1SCL	I/O	OD	I ² C module 1 clock.
	PWM0	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	U1CTS	I	TTL	UART module 1 Clear To Send modem status input signal.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
L7	RXIN	I	Analog	RXIN of the Ethernet PHY.
L8	TXON	O	TTL	TXON of the Ethernet PHY.
L9	MDIO	I/O	OD	MDIO of the Ethernet PHY.
L10	GND	-	Power	Ground reference for logic and I/O pins.
L11	OSCO	I	Analog	Main oscillator crystal input or an external clock reference input.
L12	VBAT	-	Power	Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply.

Table 24-7. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type ^a	Description
M1	PC5	I/O	TTL	GPIO port C bit 5.
	C0o	O	TTL	Analog comparator 0 output.
	C1+	I	Analog	Analog comparator 1 positive input.
	C1o	O	TTL	Analog comparator 1 output.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	Fault2	I	TTL	PWM Fault 2.
	USB0EPEN	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
M2	PC6	I/O	TTL	GPIO port C bit 6.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	PhB0	I	TTL	QEI module 0 phase B.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
M3	PA1	I/O	TTL	GPIO port A bit 1.
	I2C1SDA	I/O	OD	I ² C module 1 data.
	U0Tx	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
M4	PA2	I/O	TTL	GPIO port A bit 2.
	I2S0RXSD	I/O	TTL	I ² S module 0 receive data.
	PWM4	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	SSI0Clk	I/O	TTL	SSI module 0 clock.
M5	PA5	I/O	TTL	GPIO port A bit 5.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	I2S0TXWS	I/O	TTL	I ² S module 0 transmit word select.
	SSI0Tx	O	TTL	SSI module 0 transmit.
M6	PA7	I/O	TTL	GPIO port A bit 7.
	CAN0Tx	O	TTL	CAN module 0 transmit.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	I2C1SDA	I/O	OD	I ² C module 1 data.
	PWM1	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
	PWM5	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
	U1DCD	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
	USB0PFLT	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
M7	RXIP	I	Analog	RXIP of the Ethernet PHY.
M8	TXOP	O	TTL	TXOP of the Ethernet PHY.

Table 24-7. Signals by Pin Number (continued)

Pin Number	Pin Name	Pin Type	Buffer Type ^a	Description
M9	PF0	I/O	TTL	GPIO port F bit 0.
	CAN1Rx	I	TTL	CAN module 1 receive.
	I2S0TXSD	I/O	TTL	I ² S module 0 transmit data.
	PWM0	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
	PhB0	I	TTL	QEI module 0 phase B.
	U1DSR	I	TTL	UART module 1 Data Set Ready modem output control line.
M10	WAKE	I	TTL	An external input that brings the processor out of Hibernate mode when asserted.
M11	OSC1	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
M12	HIB	O	OD	An open-drain output that indicates the processor is in Hibernate mode.

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

Table 24-8. Signals by Signal Name

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
AIN0	B1	PE7	I	Analog	Analog-to-digital converter input 0.
AIN1	A1	PE6	I	Analog	Analog-to-digital converter input 1.
AIN2	B3	PE5	I	Analog	Analog-to-digital converter input 2.
AIN3	B2	PE4	I	Analog	Analog-to-digital converter input 3.
AIN4	A2	PD7	I	Analog	Analog-to-digital converter input 4.
AIN5	A3	PD6	I	Analog	Analog-to-digital converter input 5.
AIN6	C6	PD5	I	Analog	Analog-to-digital converter input 6.
AIN7	B5	PD4	I	Analog	Analog-to-digital converter input 7.
AIN8	B4	PE3	I	Analog	Analog-to-digital converter input 8.
AIN9	A4	PE2	I	Analog	Analog-to-digital converter input 9.
AIN10	A6	PB4	I	Analog	Analog-to-digital converter input 10.
AIN11	B7	PB5	I	Analog	Analog-to-digital converter input 11.
AIN12	H1	PD3	I	Analog	Analog-to-digital converter input 12.
AIN13	H2	PD2	I	Analog	Analog-to-digital converter input 13.
AIN14	G2	PD1	I	Analog	Analog-to-digital converter input 14.
AIN15	G1	PD0	I	Analog	Analog-to-digital converter input 15.
C0+	A7	PB6	I	Analog	Analog comparator 0 positive input.
C0-	A6	PB4	I	Analog	Analog comparator 0 negative input.
C0o	M1 K4 A7 B7 A2	PC5 (3) PF4 (2) PB6 (3) PB5 (1) PD7 (2)	O	TTL	Analog comparator 0 output.
C1+	M1	PC5	I	Analog	Analog comparator 1 positive input.
C1-	B7	PB5	I	Analog	Analog comparator 1 negative input.

Table 24-8. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
C1o	A1 L2 M1 K3 D11	PE6 (2) PC7 (7) PC5 (2) PF5 (2) PH2 (2)	O	TTL	Analog comparator 1 output.
CAN0Rx	G1 L5 L6 A6	PD0 (2) PA4 (5) PA6 (6) PB4 (5)	I	TTL	CAN module 0 receive.
CAN0Tx	G2 M5 M6 B7	PD1 (2) PA5 (5) PA7 (6) PB5 (5)	O	TTL	CAN module 0 transmit.
CAN1Rx	M9	PF0 (1)	I	TTL	CAN module 1 receive.
CAN1Tx	H12	PF1 (1)	O	TTL	CAN module 1 transmit.
CCP0	H1 L2 M2 K6 K4 E12 A11 B7 B5	PD3 (4) PC7 (4) PC6 (6) PJ2 (9) PF4 (1) PB0 (1) PB2 (5) PB5 (4) PD4 (1)	I/O	TTL	Capture/Compare/PWM 0.
CCP1	M1 L1 L6 D12 A7 B4 A2	PC5 (1) PC4 (9) PA6 (2) PB1 (4) PB6 (1) PE3 (1) PD7 (3)	I/O	TTL	Capture/Compare/PWM 1.
CCP2	B2 G2 L1 K3 D12 A12 B7 A4 C6	PE4 (6) PD1 (10) PC4 (5) PF5 (1) PB1 (1) PE1 (4) PB5 (6) PE2 (5) PD5 (1)	I/O	TTL	Capture/Compare/PWM 2.
CCP3	B2 M2 M1 M6 H12 A11 B11 B5	PE4 (1) PC6 (1) PC5 (5) PA7 (7) PF1 (10) PB2 (4) PE0 (3) PD4 (2)	I/O	TTL	Capture/Compare/PWM 3.
CCP4	L2 L1 M6 A4 C6	PC7 (1) PC4 (6) PA7 (2) PE2 (1) PD5 (2)	I/O	TTL	Capture/Compare/PWM 4.

Table 24-8. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
CCP5	B3 H2 L1 C10 A7 B7	PE5 (1) PD2 (4) PC4 (1) PG7 (8) PB6 (6) PB5 (2)	I/O	TTL	Capture/Compare/PWM 5.
CCP6	G1 H2 A12 C9 B7	PD0 (6) PD2 (2) PE1 (5) PH0 (1) PB5 (3)	I/O	TTL	Capture/Compare/PWM 6.
CCP7	G2 H1 C8 A7 B4	PD1 (6) PD3 (2) PH1 (1) PB6 (2) PE3 (5)	I/O	TTL	Capture/Compare/PWM 7.
ERBIAS	J3	fixed	O	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
Fault0	B2 K6 K4 E11 A12 D10 A3	PE4 (4) PJ2 (10) PF4 (4) PB3 (2) PE1 (3) PH3 (2) PD6 (1)	I	TTL	PWM Fault 0.
Fault1	A7	PB6 (4)	I	TTL	PWM Fault 1.
Fault2	M1 F10	PC5 (4) PH5 (10)	I	TTL	PWM Fault 2.
Fault3	E11 D11	PB3 (4) PH2 (4)	I	TTL	PWM Fault 3.
GND	C4 C5 K5 L10 K10 J10 F11 F12	fixed	-	Power	Ground reference for logic and I/O pins.
GNDA	A5	fixed	-	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.
<u>HIB</u>	M12	fixed	O	OD	An open-drain output that indicates the processor is in Hibernate mode.
I ² C0SCL	A11	PB2 (1)	I/O	OD	I ² C module 0 clock.
I ² C0SDA	E11	PB3 (1)	I/O	OD	I ² C module 0 data.
I ² C1SCL	F3 K1 L3 L6	PJ0 (11) PG0 (3) PA0 (8) PA6 (1)	I/O	OD	I ² C module 1 clock.

Table 24-8. Signals by Signal Name (continued)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
I2C1SDA	K2 M3 M6 B6	PG1 (3) PA1 (8) PA7 (1) PJ1 (11)	I/O	OD	I ² C module 1 data.
I2S0RXMCLK	L4 C6	PA3 (9) PD5 (8)	I/O	TTL	I ² S module 0 receive master clock.
I2S0RXSCK	G1	PD0 (8)	I/O	TTL	I ² S module 0 receive clock.
I2S0RXSD	M4 B5	PA2 (9) PD4 (8)	I/O	TTL	I ² S module 0 receive data.
I2S0RXWS	G2	PD1 (8)	I/O	TTL	I ² S module 0 receive word select.
I2S0TXMCLK	H12	PF1 (8)	I/O	TTL	I ² S module 0 transmit master clock.
I2S0TXSCK	L5 A7 A3	PA4 (9) PB6 (9) PD6 (8)	I/O	TTL	I ² S module 0 transmit clock.
I2S0TXSD	B3 M9	PE5 (9) PF0 (8)	I/O	TTL	I ² S module 0 transmit data.
I2S0TXWS	B2 M5 A2	PE4 (9) PA5 (9) PD7 (8)	I/O	TTL	I ² S module 0 transmit word select.
IDX0	G1 A11 A7 A6 A2	PD0 (3) PB2 (2) PB6 (5) PB4 (6) PD7 (1)	I	TTL	QEI module 0 index.
IDX1	H12 D11	PF1 (2) PH2 (1)	I	TTL	QEI module 1 index.
LDO	E3	fixed	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 μ 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	J12	PF3 (1)	O	TTL	Ethernet LED 0.
LED1	J11	PF2 (1)	O	TTL	Ethernet LED 1.
MDIO	L9	fixed	I/O	OD	MDIO of the Ethernet PHY.
NC	C1 C2 D2 D1 E1 E2 F1 F2	fixed	-	-	No connect. Leave the pin electrically unconnected/isolated.
NMI	A8	PB7 (4)	I	TTL	Non-maskable interrupt.
OSC0	L11	fixed	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	M11	fixed	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
PA0	L3	-	I/O	TTL	GPIO port A bit 0.

Table 24-8. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
PA1	M3	-	I/O	TTL	GPIO port A bit 1.
PA2	M4	-	I/O	TTL	GPIO port A bit 2.
PA3	L4	-	I/O	TTL	GPIO port A bit 3.
PA4	L5	-	I/O	TTL	GPIO port A bit 4.
PA5	M5	-	I/O	TTL	GPIO port A bit 5.
PA6	L6	-	I/O	TTL	GPIO port A bit 6.
PA7	M6	-	I/O	TTL	GPIO port A bit 7.
PB0	E12	-	I/O	TTL	GPIO port B bit 0.
PB1	D12	-	I/O	TTL	GPIO port B bit 1.
PB2	A11	-	I/O	TTL	GPIO port B bit 2.
PB3	E11	-	I/O	TTL	GPIO port B bit 3.
PB4	A6	-	I/O	TTL	GPIO port B bit 4.
PB5	B7	-	I/O	TTL	GPIO port B bit 5.
PB6	A7	-	I/O	TTL	GPIO port B bit 6.
PB7	A8	-	I/O	TTL	GPIO port B bit 7.
PC0	A9	-	I/O	TTL	GPIO port C bit 0.
PC1	B9	-	I/O	TTL	GPIO port C bit 1.
PC2	B8	-	I/O	TTL	GPIO port C bit 2.
PC3	A10	-	I/O	TTL	GPIO port C bit 3.
PC4	L1	-	I/O	TTL	GPIO port C bit 4.
PC5	M1	-	I/O	TTL	GPIO port C bit 5.
PC6	M2	-	I/O	TTL	GPIO port C bit 6.
PC7	L2	-	I/O	TTL	GPIO port C bit 7.
PD0	G1	-	I/O	TTL	GPIO port D bit 0.
PD1	G2	-	I/O	TTL	GPIO port D bit 1.
PD2	H2	-	I/O	TTL	GPIO port D bit 2.
PD3	H1	-	I/O	TTL	GPIO port D bit 3.
PD4	B5	-	I/O	TTL	GPIO port D bit 4.
PD5	C6	-	I/O	TTL	GPIO port D bit 5.
PD6	A3	-	I/O	TTL	GPIO port D bit 6.
PD7	A2	-	I/O	TTL	GPIO port D bit 7.
PE0	B11	-	I/O	TTL	GPIO port E bit 0.
PE1	A12	-	I/O	TTL	GPIO port E bit 1.
PE2	A4	-	I/O	TTL	GPIO port E bit 2.
PE3	B4	-	I/O	TTL	GPIO port E bit 3.
PE4	B2	-	I/O	TTL	GPIO port E bit 4.
PE5	B3	-	I/O	TTL	GPIO port E bit 5.
PE6	A1	-	I/O	TTL	GPIO port E bit 6.
PE7	B1	-	I/O	TTL	GPIO port E bit 7.
PF0	M9	-	I/O	TTL	GPIO port F bit 0.
PF1	H12	-	I/O	TTL	GPIO port F bit 1.

Table 24-8. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
PF2	J11	-	I/O	TTL	GPIO port F bit 2.
PF3	J12	-	I/O	TTL	GPIO port F bit 3.
PF4	K4	-	I/O	TTL	GPIO port F bit 4.
PF5	K3	-	I/O	TTL	GPIO port F bit 5.
PG0	K1	-	I/O	TTL	GPIO port G bit 0.
PG1	K2	-	I/O	TTL	GPIO port G bit 1.
PG7	C10	-	I/O	TTL	GPIO port G bit 7.
PH0	C9	-	I/O	TTL	GPIO port H bit 0.
PH1	C8	-	I/O	TTL	GPIO port H bit 1.
PH2	D11	-	I/O	TTL	GPIO port H bit 2.
PH3	D10	-	I/O	TTL	GPIO port H bit 3.
PH4	B10	-	I/O	TTL	GPIO port H bit 4.
PH5	F10	-	I/O	TTL	GPIO port H bit 5.
PH6	G3	-	I/O	TTL	GPIO port H bit 6.
PH7	H3	-	I/O	TTL	GPIO port H bit 7.
PhA0	G2 L1 A4	PD1 (3) PC4 (2) PE2 (4)	I	TTL	QEI module 0 phase A.
PhA1	B4	PE3 (3)	I	TTL	QEI module 1 phase A.
PhB0	L2 M2 M9 D10 B4	PC7 (2) PC6 (2) PF0 (2) PH3 (1) PE3 (4)	I	TTL	QEI module 0 phase B.
PhB1	G2 C10 A4	PD1 (11) PG7 (1) PE2 (3)	I	TTL	QEI module 1 phase B.
PJ0	F3	-	I/O	TTL	GPIO port J bit 0.
PJ1	B6	-	I/O	TTL	GPIO port J bit 1.
PJ2	K6	-	I/O	TTL	GPIO port J bit 2.
PWM0	G1 F3 K1 L6 M9	PD0 (1) PJ0 (10) PG0 (2) PA6 (4) PF0 (3)	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
PWM1	G2 K2 M6 H12 B6	PD1 (1) PG1 (2) PA7 (4) PF1 (3) PJ1 (10)	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
PWM2	H2 J11 E12 C9	PD2 (3) PF2 (4) PB0 (2) PH0 (2)	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.

Table 24-8. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
PWM3	H1 J12 D12 C8	PD3 (3) PF3 (4) PB1 (2) PH1 (2)	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
PWM4	A1 K1 M4 L6 J11 G3 B11 C9	PE6 (1) PG0 (4) PA2 (4) PA6 (5) PF2 (2) PH6 (10) PE0 (1) PH0 (9)	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
PWM5	B1 H3 K2 L4 M6 J12 A12 C8	PE7 (1) PH7 (10) PG1 (4) PA3 (4) PA7 (5) PF3 (2) PE1 (1) PH1 (9)	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.
RST	H11	fixed	I	TTL	System reset input.
RXIN	L7	fixed	I	Analog	RXIN of the Ethernet PHY.
RXIP	M7	fixed	I	Analog	RXIP of the Ethernet PHY.
SSI0Clk	M4	PA2 (1)	I/O	TTL	SSI module 0 clock.
SSI0Fss	L4	PA3 (1)	I/O	TTL	SSI module 0 frame.
SSI0Rx	L5	PA4 (1)	I	TTL	SSI module 0 receive.
SSI0Tx	M5	PA5 (1)	O	TTL	SSI module 0 transmit.
SSI1Clk	J11 B11 B10	PF2 (9) PE0 (2) PH4 (11)	I/O	TTL	SSI module 1 clock.
SSI1Fss	J12 F10 A12	PF3 (9) PH5 (11) PE1 (2)	I/O	TTL	SSI module 1 frame.
SSI1Rx	K4 G3 A4	PF4 (9) PH6 (11) PE2 (2)	I	TTL	SSI module 1 receive.
SSI1Tx	H3 K3 B4	PH7 (11) PF5 (9) PE3 (2)	O	TTL	SSI module 1 transmit.
SWCLK	A9	PC0 (3)	I	TTL	JTAG/SWD CLK.
SWDIO	B9	PC1 (3)	I/O	TTL	JTAG TMS and SWDIO.
SWO	A10	PC3 (3)	O	TTL	JTAG TDO and SWO.
TCK	A9	PC0 (3)	I	TTL	JTAG/SWD CLK.
TDI	B8	PC2 (3)	I	TTL	JTAG TDI.
TDO	A10	PC3 (3)	O	TTL	JTAG TDO and SWO.
TMS	B9	PC1 (3)	I	TTL	JTAG TMS and SWDIO.
TXON	L8	fixed	O	TTL	TXON of the Ethernet PHY.
TXOP	M8	fixed	O	TTL	TXOP of the Ethernet PHY.

Table 24-8. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
U0Rx	L3	PA0 (1)	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
U0Tx	M3	PA1 (1)	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
U1CTS	A1 G1 L6	PE6 (9) PD0 (9) PA6 (9)	I	TTL	UART module 1 Clear To Send modem status input signal.
U1DCD	B1 G2 M6	PE7 (9) PD1 (9) PA7 (9)	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
U1DSR	M9	PF0 (9)	I	TTL	UART module 1 Data Set Ready modem output control line.
U1DTR	A2	PD7 (9)	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
U1RI	B5	PD4 (9)	I	TTL	UART module 1 Ring Indicator modem status input signal.
U1RTS	H12	PF1 (9)	O	TTL	UART module 1 Request to Send modem output control line.
U1Rx	G1 H2 M2 L3 E12 A6	PD0 (5) PD2 (1) PC6 (5) PA0 (9) PB0 (5) PB4 (7)	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
U1Tx	G2 H1 L2 M3 D12 B7	PD1 (5) PD3 (1) PC7 (5) PA1 (9) PB1 (5) PB5 (7)	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
U2Rx	G1 K1 A6 C6	PD0 (4) PG0 (1) PB4 (4) PD5 (9)	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
U2Tx	B2 G2 K2 A3	PE4 (5) PD1 (4) PG1 (1) PD6 (9)	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
USB0DM	C11	fixed	I/O	Analog	Bidirectional differential data pin (D- per USB specification).
USB0DP	C12	fixed	I/O	Analog	Bidirectional differential data pin (D+ per USB specification).
USB0EPEN	K1 M1 L6 A11 D10	PG0 (7) PC5 (6) PA6 (8) PB2 (8) PH3 (4)	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
USB0ID	E12	PB0	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 the A side of the cable and pulled up is the B side).

Table 24-8. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
USB0PFLT	L2 M2 M6 E11 B11 B10 B6	PC7 (6) PC6 (7) PA7 (8) PB3 (8) PE0 (9) PH4 (4) PJ1 (9)	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
USB0RBIAST	B12	fixed	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.
USB0VBUS	D12	PB1	I/O	Analog	This signal is used during the session request protocol. This signal allows the USB PHY to both sense the voltage level of VBUS, and pull up VBUS momentarily during VBUS pulsing.
VBAT	L12	fixed	-	Power	Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply.
VDD	K7 G12 K8 K9 H10 G10 E10 G11	fixed	-	Power	Positive supply for I/O and some logic.
VDDA	C7	fixed	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be connected to 3.3 V, regardless of system implementation.
VDDC	D3 C3	fixed	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VREFA	A7	PB6	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 AINn signal is converted to 1023. The VREFA input is limited to the range specified in Table 26-25 on page 1244.
WAKE	M10	fixed	I	TTL	An external input that brings the processor out of Hibernate mode when asserted.
XOSC0	K11	fixed	I	Analog	Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the HIBCTL register.
XOSC1	K12	fixed	O	Analog	Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source.
XTALNPHY	J1	fixed	O	Analog	Ethernet PHY XTALN 25-MHz oscillator crystal output.

Table 24-8. Signals by Signal Name (*continued*)

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type ^a	Description
XTALPPHY	J2	fixed	I	Analog	Ethernet PHY XTALP 25-MHz oscillator crystal input.

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

Table 24-9. Signals by Function, Except for GPIO

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	Description
ADC	AIN0	B1	I	Analog	Analog-to-digital converter input 0.
	AIN1	A1	I	Analog	Analog-to-digital converter input 1.
	AIN2	B3	I	Analog	Analog-to-digital converter input 2.
	AIN3	B2	I	Analog	Analog-to-digital converter input 3.
	AIN4	A2	I	Analog	Analog-to-digital converter input 4.
	AIN5	A3	I	Analog	Analog-to-digital converter input 5.
	AIN6	C6	I	Analog	Analog-to-digital converter input 6.
	AIN7	B5	I	Analog	Analog-to-digital converter input 7.
	AIN8	B4	I	Analog	Analog-to-digital converter input 8.
	AIN9	A4	I	Analog	Analog-to-digital converter input 9.
	AIN10	A6	I	Analog	Analog-to-digital converter input 10.
	AIN11	B7	I	Analog	Analog-to-digital converter input 11.
	AIN12	H1	I	Analog	Analog-to-digital converter input 12.
	AIN13	H2	I	Analog	Analog-to-digital converter input 13.
	AIN14	G2	I	Analog	Analog-to-digital converter input 14.
	AIN15	G1	I	Analog	Analog-to-digital converter input 15.
Analog Comparators	VREFA	A7	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 AINn signal is converted to 1023. The VREFA input is limited to the range specified in Table 26-25 on page 1244.
	C0+	A7	I	Analog	Analog comparator 0 positive input.
	C0-	A6	I	Analog	Analog comparator 0 negative input.
	C0o	M1 K4 A7 B7 A2	O	TTL	Analog comparator 0 output.
	C1+	M1	I	Analog	Analog comparator 1 positive input.
	C1-	B7	I	Analog	Analog comparator 1 negative input.
	C1o	A1 L2 M1 K3 D11	O	TTL	Analog comparator 1 output.

Table 24-9. Signals by Function, Except for GPIO (*continued*)

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	Description
Controller Area Network	CAN0Rx	G1 L5 L6 A6	I	TTL	CAN module 0 receive.
	CAN0Tx	G2 M5 M6 B7	O	TTL	CAN module 0 transmit.
	CAN1Rx	M9	I	TTL	CAN module 1 receive.
	CAN1Tx	H12	O	TTL	CAN module 1 transmit.
Ethernet	ERBIAS	J3	O	Analog	12.4-kΩ resistor (1% precision) used internally for Ethernet PHY.
	LED0	J12	O	TTL	Ethernet LED 0.
	LED1	J11	O	TTL	Ethernet LED 1.
	MDIO	L9	I/O	OD	MDIO of the Ethernet PHY.
	RXIN	L7	I	Analog	RXIN of the Ethernet PHY.
	RXIP	M7	I	Analog	RXIP of the Ethernet PHY.
	TXON	L8	O	TTL	TXON of the Ethernet PHY.
	TXOP	M8	O	TTL	TXOP of the Ethernet PHY.
	XTALNPHY	J1	O	Analog	Ethernet PHY XTALN 25-MHz oscillator crystal output.
	XTALPPHY	J2	I	Analog	Ethernet PHY XTALP 25-MHz oscillator crystal input.

Table 24-9. Signals by Function, Except for GPIO (*continued*)

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	Description
General-Purpose Timers	CCP0	H1 L2 M2 K6 K4 E12 A11 B7 B5	I/O	TTL	Capture/Compare/PWM 0.
	CCP1	M1 L1 L6 D12 A7 B4 A2	I/O	TTL	Capture/Compare/PWM 1.
	CCP2	B2 G2 L1 K3 D12 A12 B7 A4 C6	I/O	TTL	Capture/Compare/PWM 2.
	CCP3	B2 M2 M1 M6 H12 A11 B11 B5	I/O	TTL	Capture/Compare/PWM 3.
	CCP4	L2 L1 M6 A4 C6	I/O	TTL	Capture/Compare/PWM 4.
	CCP5	B3 H2 L1 C10 A7 B7	I/O	TTL	Capture/Compare/PWM 5.
	CCP6	G1 H2 A12 C9 B7	I/O	TTL	Capture/Compare/PWM 6.
	CCP7	G2 H1 C8 A7 B4	I/O	TTL	Capture/Compare/PWM 7.

Table 24-9. Signals by Function, Except for GPIO (continued)

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	Description
Hibernate	HIB	M12	O	OD	An open-drain output that indicates the processor is in Hibernate mode.
	VBAT	L12	-	Power	Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply.
	WAKE	M10	I	TTL	An external input that brings the processor out of Hibernate mode when asserted.
	XOSC0	K11	I	Analog	Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the CLKSEL bit in the HIBCTL register.
	XOSC1	K12	O	Analog	Hibernation module oscillator crystal output. Leave unconnected when using a single-ended clock source.
I ² C	I ² C0SCL	A11	I/O	OD	I ² C module 0 clock.
	I ² C0SDA	E11	I/O	OD	I ² C module 0 data.
	I ² C1SCL	F3 K1 L3 L6	I/O	OD	I ² C module 1 clock.
	I ² C1SDA	K2 M3 M6 B6	I/O	OD	I ² C module 1 data.
I ² S	I ² S0RXMCLK	L4 C6	I/O	TTL	I ² S module 0 receive master clock.
	I ² S0RXSCK	G1	I/O	TTL	I ² S module 0 receive clock.
	I ² S0RXSD	M4 B5	I/O	TTL	I ² S module 0 receive data.
	I ² S0RXWS	G2	I/O	TTL	I ² S module 0 receive word select.
	I ² S0TXMCLK	H12	I/O	TTL	I ² S module 0 transmit master clock.
	I ² S0TXSCK	L5 A7 A3	I/O	TTL	I ² S module 0 transmit clock.
	I ² S0TXSD	B3 M9	I/O	TTL	I ² S module 0 transmit data.
	I ² S0TXWS	B2 M5 A2	I/O	TTL	I ² S module 0 transmit word select.
JTAG/SWD/SWO	SWCLK	A9	I	TTL	JTAG/SWD CLK.
	SWDIO	B9	I/O	TTL	JTAG TMS and SWDIO.
	SWO	A10	O	TTL	JTAG TDO and SWO.
	TCK	A9	I	TTL	JTAG/SWD CLK.
	TDI	B8	I	TTL	JTAG TDI.
	TDO	A10	O	TTL	JTAG TDO and SWO.
	TMS	B9	I	TTL	JTAG TMS and SWDIO.

Table 24-9. Signals by Function, Except for GPIO (*continued*)

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	Description
PWM	Fault0	B2 K6 K4 E11 A12 D10 A3	I	TTL	PWM Fault 0.
	Fault1	A7	I	TTL	PWM Fault 1.
	Fault2	M1 F10	I	TTL	PWM Fault 2.
	Fault3	E11 D11	I	TTL	PWM Fault 3.
	PWM0	G1 F3 K1 L6 M9	O	TTL	PWM 0. This signal is controlled by PWM Generator 0.
	PWM1	G2 K2 M6 H12 B6	O	TTL	PWM 1. This signal is controlled by PWM Generator 0.
	PWM2	H2 J11 E12 C9	O	TTL	PWM 2. This signal is controlled by PWM Generator 1.
	PWM3	H1 J12 D12 C8	O	TTL	PWM 3. This signal is controlled by PWM Generator 1.
	PWM4	A1 K1 M4 L6 J11 G3 B11 C9	O	TTL	PWM 4. This signal is controlled by PWM Generator 2.
	PWM5	B1 H3 K2 L4 M6 J12 A12 C8	O	TTL	PWM 5. This signal is controlled by PWM Generator 2.

Table 24-9. Signals by Function, Except for GPIO (*continued*)

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	Description
Power	GND	C4 C5 K5 L10 K10 J10 F11 F12	-	Power	Ground reference for logic and I/O pins.
	GNDA	A5	-	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	E3	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 μ 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	K7 G12 K8 K9 H10 G10 E10 G11	-	Power	Positive supply for I/O and some logic.
	VDDA	C7	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions. VDDA pins must be connected to 3.3 V, regardless of system implementation.
	VDDC	D3 C3	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
QEI	IDX0	G1 A11 A7 A6 A2	I	TTL	QEI module 0 index.
	IDX1	H12 D11	I	TTL	QEI module 1 index.
	PhA0	G2 L1 A4	I	TTL	QEI module 0 phase A.
	PhA1	B4	I	TTL	QEI module 1 phase A.
	PhB0	L2 M2 M9 D10 B4	I	TTL	QEI module 0 phase B.
	PhB1	G2 C10 A4	I	TTL	QEI module 1 phase B.

Table 24-9. Signals by Function, Except for GPIO (continued)

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	Description
SSI	SSI0Clk	M4	I/O	TTL	SSI module 0 clock.
	SSI0Fss	L4	I/O	TTL	SSI module 0 frame.
	SSI0Rx	L5	I	TTL	SSI module 0 receive.
	SSI0Tx	M5	O	TTL	SSI module 0 transmit.
	SSI1Clk	J11 B11 B10	I/O	TTL	SSI module 1 clock.
	SSI1Fss	J12 F10 A12	I/O	TTL	SSI module 1 frame.
	SSI1Rx	K4 G3 A4	I	TTL	SSI module 1 receive.
	SSI1Tx	H3 K3 B4	O	TTL	SSI module 1 transmit.
System Control & Clocks	NMI	A8	I	TTL	Non-maskable interrupt.
	OSC0	L11	I	Analog	Main oscillator crystal input or an external clock reference input.
	OSC1	M11	O	Analog	Main oscillator crystal output. Leave unconnected when using a single-ended clock source.
	RST	H11	I	TTL	System reset input.

Table 24-9. Signals by Function, Except for GPIO (*continued*)

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	Description
UART	U0Rx	L3	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
	U0Tx	M3	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
	U1CTS	A1 G1 L6	I	TTL	UART module 1 Clear To Send modem status input signal.
	U1DCD	B1 G2 M6	I	TTL	UART module 1 Data Carrier Detect modem status input signal.
	U1DSR	M9	I	TTL	UART module 1 Data Set Ready modem output control line.
	U1DTR	A2	O	TTL	UART module 1 Data Terminal Ready modem status input signal.
	U1RI	B5	I	TTL	UART module 1 Ring Indicator modem status input signal.
	U1RTS	H12	O	TTL	UART module 1 Request to Send modem output control line.
	U1Rx	G1 H2 M2 L3 E12 A6	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U1Tx	G2 H1 L2 M3 D12 B7	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	U2Rx	G1 K1 A6 C6	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
	U2Tx	B2 G2 K2 A3	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.

Table 24-9. Signals by Function, Except for GPIO (continued)

Function	Pin Name	Pin Number	Pin Type	Buffer Type ^a	Description
USB	USB0DM	C11	I/O	Analog	Bidirectional differential data pin (D- per USB specification).
	USB0DP	C12	I/O	Analog	Bidirectional differential data pin (D+ per USB specification).
	USB0EPEN	K1 M1 L6 A11 D10	O	TTL	Optionally used in Host mode to control an external power source to supply power to the USB bus.
	USB0ID	E12	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 the A side of the cable and pulled up is the B side).
	USB0PFLT	L2 M2 M6 E11 B11 B10 B6	I	TTL	Optionally used in Host mode by an external power source to indicate an error state by that power source.
	USB0RBIAS	B12	O	Analog	9.1-kΩ resistor (1% precision) used internally for USB analog circuitry.
	USB0VBUS	D12	I/O	Analog	This signal is used during the session request 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 has TTL-compatible voltage levels.

Table 24-10. GPIO Pins and Alternate Functions

IO	Pin	Analog Function	Digital Function (GPIOCTL PMCx Bit Field Encoding) ^a										
			1	2	3	4	5	6	7	8	9	10	11
PA0	L3	-	U0Rx	-	-	-	-	-	-	I2C1SCL	U1Rx	-	-
PA1	M3	-	U0Tx	-	-	-	-	-	-	I2C1SDA	U1Tx	-	-
PA2	M4	-	SSI0Clk	-	-	PWM4	-	-	-	-	I2S0RXSD	-	-
PA3	L4	-	SSI0Fss	-	-	PWM5	-	-	-	-	I2S0RMTLK	-	-
PA4	L5	-	SSI0RX	-	-	-	CAN0Rx	-	-	-	I2S0TXSCK	-	-
PA5	M5	-	SSI0Tx	-	-	-	CAN0Tx	-	-	-	I2S0TXWS	-	-
PA6	L6	-	I2C1SCL	CCP1	-	PWM0	PWM4	CAN0Rx	-	USB0EPEN	U1CTS	-	-
PA7	M6	-	I2C1SDA	CCP4	-	PWM1	PWM5	CAN0Tx	CCP3	USB0PFLT	U1DCD	-	-
PB0	E12	USB0ID	CCP0	PWM2	-	-	U1Rx	-	-	-	-	-	-
PB1	D12	USB0VBUS	CCP2	PWM3	-	CCP1	U1Tx	-	-	-	-	-	-
PB2	A11	-	I2C0SCL	IDX0	-	CCP3	CCP0	-	-	USB0EPEN	-	-	-
PB3	E11	-	I2C0SDA	Fault0	-	Fault3	-	-	-	USB0PFLT	-	-	-
PB4	A6	AIN10 C0-	-	-	-	U2Rx	CAN0Rx	IDX0	U1Rx	-	-	-	-
PB5	B7	AIN11 C1-	C0o	CCP5	CCP6	CCP0	CAN0Tx	CCP2	U1Tx	-	-	-	-

Table 24-10. GPIO Pins and Alternate Functions (*continued*)

IO	Pin	Analog Function	Digital Function (GPIO PCTL PMCx Bit Field Encoding) ^a										
			1	2	3	4	5	6	7	8	9	10	11
PB6	A7	VREFA C0+	CCP1	CCP7	C0o	Fault1	IDX0	CCP5	-	-	I2S0TXSCK	-	-
PB7	A8	-	-	-	-	NMI	-	-	-	-	-	-	-
PC0	A9	-	-	-	TCK SWCLK	-	-	-	-	-	-	-	-
PC1	B9	-	-	-	TMS SWDIO	-	-	-	-	-	-	-	-
PC2	B8	-	-	-	TDI	-	-	-	-	-	-	-	-
PC3	A10	-	-	-	TDO SWO	-	-	-	-	-	-	-	-
PC4	L1	-	CCP5	PhA0	-	-	CCP2	CCP4	-	-	CCP1	-	-
PC5	M1	C1+	CCP1	C1o	C0o	Fault2	CCP3	USB0EPEN	-	-	-	-	-
PC6	M2	-	CCP3	PhB0	-	-	U1Rx	CCP0	USB0PFLT	-	-	-	-
PC7	L2	-	CCP4	PhB0	-	CCP0	U1Tx	USB0PFLT	C1o	-	-	-	-
PD0	G1	AIN15	PWM0	CAN0Rx	IDX0	U2Rx	U1Rx	CCP6	-	I2S0RXSCK	U1CTS	-	-
PD1	G2	AIN14	PWM1	CAN0Tx	PhA0	U2Tx	U1Tx	CCP7	-	I2S0RXWS	U1DCD	CCP2	PhB1
PD2	H2	AIN13	U1Rx	CCP6	PWM2	CCP5	-	-	-	-	-	-	-
PD3	H1	AIN12	U1Tx	CCP7	PWM3	CCP0	-	-	-	-	-	-	-
PD4	B5	AIN7	CCP0	CCP3	-	-	-	-	-	I2S0RXSD	U1RI	-	-
PD5	C6	AIN6	CCP2	CCP4	-	-	-	-	-	I2S0RMCLK	U2Rx	-	-
PD6	A3	AIN5	Fault0	-	-	-	-	-	-	I2S0TXSCK	U2Tx	-	-
PD7	A2	AIN4	IDX0	C0o	CCP1	-	-	-	-	I2S0TXWS	U1DTR	-	-
PE0	B11	-	PWM4	SSI1Clk	CCP3	-	-	-	-	-	USB0PFLT	-	-
PE1	A12	-	PWM5	SSI1Fss	Fault0	CCP2	CCP6	-	-	-	-	-	-
PE2	A4	AIN9	CCP4	SSI1Rx	PhB1	PhA0	CCP2	-	-	-	-	-	-
PE3	B4	AIN8	CCP1	SSI1Tx	PhA1	PhB0	CCP7	-	-	-	-	-	-
PE4	B2	AIN3	CCP3	-	-	Fault0	U2Tx	CCP2	-	-	I2S0TXWS	-	-
PE5	B3	AIN2	CCP5	-	-	-	-	-	-	-	I2S0TXSD	-	-
PE6	A1	AIN1	PWM4	C1o	-	-	-	-	-	-	U1CTS	-	-
PE7	B1	AIN0	PWM5	-	-	-	-	-	-	-	U1DCD	-	-
PF0	M9	-	CAN1Rx	PhB0	PWM0	-	-	-	-	I2S0TXSD	U1DSR	-	-
PF1	H12	-	CAN1Tx	IDX1	PWM1	-	-	-	-	I2S0TXMCLK	U1RTS	CCP3	-
PF2	J11	-	LED1	PWM4	-	PWM2	-	-	-	-	SSI1Clk	-	-
PF3	J12	-	LED0	PWM5	-	PWM3	-	-	-	-	SSI1Fss	-	-
PF4	K4	-	CCP0	C0o	-	Fault0	-	-	-	-	SSI1Rx	-	-
PF5	K3	-	CCP2	C1o	-	-	-	-	-	-	SSI1Tx	-	-
PG0	K1	-	U2Rx	PWM0	I2C1SCL	PWM4	-	-	USBOEPEN	-	-	-	-
PG1	K2	-	U2Tx	PWM1	I2C1SDA	PWM5	-	-	-	-	-	-	-
PG7	C10	-	PhB1	-	-	-	-	-	-	CCP5	-	-	-
PH0	C9	-	CCP6	PWM2	-	-	-	-	-	-	PWM4	-	-
PH1	C8	-	CCP7	PWM3	-	-	-	-	-	-	PWM5	-	-
PH2	D11	-	IDX1	C1o	-	Fault3	-	-	-	-	-	-	-

Table 24-10. GPIO Pins and Alternate Functions (*continued*)

IO	Pin	Analog Function	Digital Function (GPIO PCTL PMCx Bit Field Encoding) ^a										
			1	2	3	4	5	6	7	8	9	10	11
PH3	D10	-	PhB0	Fault0	-	USB0EPEN	-	-	-	-	-	-	-
PH4	B10	-	-	-	-	USB0PFLT	-	-	-	-	-	-	SSI1Clk
PH5	F10	-	-	-	-	-	-	-	-	-	-	Fault2	SSI1Fss
PH6	G3	-	-	-	-	-	-	-	-	-	-	PWM4	SSI1Rx
PH7	H3	-	-	-	-	-	-	-	-	-	-	PWM5	SSI1Tx
PJ0	F3	-	-	-	-	-	-	-	-	-	-	PWM0	I2C1SCL
PJ1	B6	-	-	-	-	-	-	-	-	-	USB0PFLT	PWM1	I2C1SDA
PJ2	K6	-	-	-	-	-	-	-	-	-	CCP0	Fault0	-

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin.

Table 24-11. Possible Pin Assignments for Alternate Functions

# of Possible Assignments	Alternate Function	GPIO Function
one	AIN0	PE7
	AIN1	PE6
	AIN10	PB4
	AIN11	PB5
	AIN12	PD3
	AIN13	PD2
	AIN14	PD1
	AIN15	PD0
	AIN2	PE5
	AIN3	PE4
	AIN4	PD7
	AIN5	PD6
	AIN6	PD5
	AIN7	PD4
	AIN8	PE3
	AIN9	PE2
	C0+	PB6
	C0-	PB4
	C1+	PC5
	C1-	PB5
	CAN1Rx	PF0
	CAN1Tx	PF1
	Fault1	PB6
	I2C0SCL	PB2
	I2C0SDA	PB3
	I2S0RXSCK	PD0
	I2S0RXWS	PD1
	I2S0TXMCLK	PF1
	LED0	PF3
	LED1	PF2
	NMI	PB7
	PhA1	PE3
	SSI0Clk	PA2
	SSI0FSS	PA3
	SSI0Rx	PA4
	SSI0Tx	PA5
	SWCLK	PC0
	SWDIO	PC1
	SWO	PC3
	TCK	PC0
	TDI	PC2

Table 24-11. Possible Pin Assignments for Alternate Functions (continued)

# of Possible Assignments	Alternate Function	GPIO Function
two	TDO	PC3
	TMS	PC1
	U0Rx	PA0
	U0Tx	PA1
	U1DSR	PF0
	U1DTR	PD7
	U1RI	PD4
	U1RTS	PF1
	USB0ID	PB0
	USB0VBUS	PB1
three	VREFA	PB6
	Fault2	PC5 PH5
	Fault3	PB3 PH2
	I2S0RXMCLK	PA3 PD5
	I2S0RXSD	PA2 PD4
	I2S0TXSD	PE5 PF0
four	IDX1	PF1 PH2
	I2S0TXSCK	PA4 PB6 PD6
	I2S0TXWS	PE4 PA5 PD7
	PhA0	PD1 PC4 PE2
	PhB1	PD1 PG7 PE2
	SSI1Clk	PF2 PE0 PH4
	SSI1FSS	PF3 PH5 PE1
	SSI1Rx	PF4 PH6 PE2
	SSI1Tx	PH7 PF5 PE3
	U1CTS	PE6 PD0 PA6
	U1DCD	PE7 PD1 PA7
six	CAN0Rx	PD0 PA4 PA6 PB4
	CAN0Tx	PD1 PA5 PA7 PB5
	I2C1SCL	PJ0 PG0 PA0 PA6
	I2C1SDA	PG1 PA1 PA7 PJ1
	PWM2	PD2 PF2 PB0 PH0
	PWM3	PD3 PF3 PB1 PH1
	U2Rx	PD0 PG0 PB4 PD5
	U2Tx	PE4 PD1 PG1 PD6

Table 24-11. Possible Pin Assignments for Alternate Functions (continued)

# of Possible Assignments	Alternate Function	GPIO Function
five	C0o	PC5 PF4 PB6 PB5 PD7
	C1o	PE6 PC7 PC5 PF5 PH2
	CCP4	PC7 PC4 PA7 PE2 PD5
	CCP6	PD0 PD2 PE1 PH0 PB5
	CCP7	PD1 PD3 PH1 PB6 PE3
	IDX0	PD0 PB2 PB6 PB4 PD7
	PWM0	PD0 PJ0 PG0 PA6 PF0
	PWM1	PD1 PG1 PA7 PF1 PJ1
	PhB0	PC7 PC6 PF0 PH3 PE3
	USB0EPEN	PG0 PC5 PA6 PB2 PH3
six	CCP5	PE5 PD2 PC4 PG7 PB6 PB5
	U1Rx	PD0 PD2 PC6 PA0 PB0 PB4
	U1Tx	PD1 PD3 PC7 PA1 PB1 PB5
seven	CCP1	PC5 PC4 PA6 PB1 PB6 PE3 PD7
	Fault0	PE4 PJ2 PF4 PB3 PE1 PH3 PD6
	USB0PFLT	PC7 PC6 PA7 PB3 PE0 PH4 PJ1
eight	CCP3	PE4 PC6 PC5 PA7 PF1 PB2 PE0 PD4
	PWM4	PE6 PG0 PA2 PA6 PF2 PH6 PE0 PH0
	PWM5	PE7 PH7 PG1 PA3 PA7 PF3 PE1 PH1
nine	CCP0	PD3 PC7 PC6 PJ2 PF4 PB0 PB2 PB5 PD4
	CCP2	PE4 PD1 PC4 PF5 PB1 PE1 PB5 PE2 PD5

24.3 Connections for Unused Signals

Table 24-12 on page 1228 show how to handle signals for functions that are not used in a particular system implementation for devices that are in a 100-pin LQFP package. Two options are shown in the table: an acceptable practice and a preferred practice for reduced power consumption and improved EMC characteristics. If a module is not used in a system, and its inputs are grounded, it is important that the clock to the module is never enabled by setting the corresponding bit in the **RCGCx** register.

Table 24-12. Connections for Unused Signals (100-pin LQFP)

Function	Signal Name	Pin Number	Acceptable Practice	Preferred Practice
Ethernet	ERBIAS	33	Connect to GND through 12.4-k Ω resistor.	Connect to GND through 12.4-k Ω resistor.
	MDIO ^a	58	NC	NC
	RXIN	37	NC	GND
	RXIP	40	NC	GND
	TXON	46	NC	GND
	TXOP	43	NC	GND
	XTALNPHY ^a	17	NC	NC
	XTALPPHY ^a	16	NC	GND
GPIO	All unused GPIOs	-	NC	GND

Table 24-12. Connections for Unused Signals (100-pin LQFP) (continued)

Function	Signal Name	Pin Number	Acceptable Practice	Preferred Practice
Hibernate	HIB	51	NC	NC
	VBAT	55	NC	GND
	WAKE	50	NC	GND
	XOSC0	52	NC	GND
	XOSC1	53	NC	NC
No Connects	NC	-	NC	NC
System Control	OSC0	48	NC	GND
	OSC1	49	NC	NC
	RST	48	Pull up as shown in Figure 5-1 on page 197	Connect through a capacitor to GND as close to pin as possible
USB	USB0RBIAS	73	Connect to GND through 10-kΩ resistor.	Connect to GND through 10-kΩ resistor.
	USB0DM	70	NC	GND
	USB0DP	71	NC	GND

a. Note that the Ethernet PHY is powered up by default. The PHY cannot be powered down unless a clock source is provided and the MDIO pin is pulled up through a 10-kΩ resistor.

Table 24-13 on page 1229 show how to handle signals for functions that are not used in a particular system implementation for devices that are in a 108-pin BGA package. Two options are shown in the table: an acceptable practice and a preferred practice for reduced power consumption and improved EMC characteristics. If a module is not used in a system, and its inputs are grounded, it is important that the clock to the module is never enabled by setting the corresponding bit in the RCGCx register.

Table 24-13. Connections for Unused Signals, 108-pin BGA

Function	Signal Name	Pin Number	Acceptable Practice	Preferred Practice
Ethernet	ERBIAS	J3	Connect to GND through 12.4-kΩ resistor.	Connect to GND through 12.4-kΩ resistor.
	MDIO ^a	L9	NC	NC
	RXIN	L7	NC	GND
	RXIP	M7	NC	GND
	TXON	L8	NC	GND
	TXOP	M8	NC	GND
	XTALNPHY ^a	J1	NC	NC
	XTALPPHY ^a	J2	NC	GND
GPIO	All unused GPIOs	-	NC	GND
Hibernate	HIB	M12	NC	NC
	VBAT	L12	NC	GND
	WAKE	M10	NC	GND
	XOSC0	K11	NC	GND
	XOSC1	K12	NC	NC
No Connects	NC	-	NC	NC

Table 24-13. Connections for Unused Signals, 108-pin BGA (*continued*)

Function	Signal Name	Pin Number	Acceptable Practice	Preferred Practice
System Control	OSC0	L11	NC	GND
	OSC1	M11	NC	NC
	$\overline{\text{RST}}$	H11	Pull up as shown in Figure 5-1 on page 197	Connect through a capacitor to GND as close to pin as possible
USB	USB0RBIAS	B12	Connect to GND through 10-k Ω resistor.	Connect to GND through 10-k Ω resistor.
	USB0DM	C11	NC	GND
	USB0DP	C12	NC	GND

a. Note that the Ethernet PHY is powered up by default. The PHY cannot be powered down unless a clock source is provided and the MDIO pin is pulled up through a 10-k Ω resistor.

25 Operating Characteristics

Table 25-1. Temperature Characteristics

Characteristic	Symbol	Value	Unit
Industrial operating temperature range	T _A	-40 to +85	°C
Unpowered storage temperature range	T _S	-65 to +150	°C

Table 25-2. Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance (junction to ambient) ^a	Θ _{JA}	32	°C/W
Junction temperature ^b	T _J	T _A + (P • Θ _{JA})	°C

a. Junction to ambient thermal resistance Θ_{JA} numbers are determined by a package simulator.

b. Power dissipation is a function of temperature.

Table 25-3. ESD Absolute Maximum Ratings^a

Parameter Name	Min	Nom	Max	Unit
V _{ESDHBM}	-	-	2.0	kV
V _{ESDCDM}	-	-	1.0	kV
V _{ESDMM}	-	-	100	V

a. All Stellaris parts are ESD tested following the JEDEC standard.

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 ^a	Value		Unit
		Min	Max	
V_{DD}	I/O supply voltage (V_{DD})	0	4	V
V_{DDA}	Analog supply voltage (V_{DDA})	0	4	V
V_{BAT}	Battery supply voltage (V_{BAT})	0	4	V
V_{IN}	Input voltage	-0.3	5.5	V
	Input voltage for a GPIO configured as an analog input	-0.3	$V_{DD} + 0.3$	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_{DD}).

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_{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 or BGA pin group 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_{DD}	I/O supply voltage	3.0	3.3	3.6	V
V_{DDA}	Analog supply voltage	3.0	3.3	3.6	V
V_{DDC}^a	Core supply voltage	1.08	1.2	1.32	V
V_{IH}	High-level input voltage	2.0	-	5.0	V
	High-level input voltage - OSC0, XOSC0, XTALPPHY single-ended clock source	$V_{DD} - 0.5$	-	3.6	V
V_{IL}	Low-level input voltage	-0.3	-	1.3	V
	Low-level input voltage - OSC0, XOSC0, XTALPPHY single-ended clock source	-0.3	-	0.5	V

Table 26-2. Recommended DC Operating Conditions (continued)

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{OH}^b	High-level output voltage	2.4	-	-	V
V_{OL}^a	Low-level output voltage	-	-	0.4	V
I_{OH}	High-level source current, $V_{OH}=2.4\text{ V}$				
	2-mA Drive	2.0	-	-	mA
	4-mA Drive	4.0	-	-	mA
	8-mA Drive	8.0	-	-	mA
I_{OL}	Low-level sink current, $V_{OL}=0.4\text{ V}$				
	2-mA Drive	2.0	-	-	mA
	4-mA Drive	4.0	-	-	mA
	8-mA Drive	8.0	-	-	mA

a. V_{DDC} is supplied from the output of the LDO.b. V_{OL} and V_{OH} 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_{LDO}	External filter capacitor size for internal power supply	1.0	-	3.0	μF
V_{LDO}	LDO output voltage	1.08	1.2	1.32	V

26.1.4 Hibernation Module Characteristics

Table 26-4. Hibernation Module DC Characteristics

Parameter	Parameter Name	Min	Nominal	Max	Unit
V_{BAT}	Battery supply voltage	2.4	3.0	3.6	V
V_{LOWBAT}	Low battery detect voltage	-	2.35	-	V

26.1.5 Flash Memory Characteristics

Table 26-5. Flash Memory Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
PE_{CYC}	Number of guaranteed mass program/erase cycles before failure ^a	15,000	-	-	cycles
T_{RET}	Data retention at average operating temperature of 125°C	10	-	-	years
T_{PROG}	Word program time	-	-	1	ms
T_{BPROG}	Buffer program time	-	-	1	ms
T_{ERASE}	Page erase time	-	-	12	ms
T_{ME}	Mass erase time	-	-	16	ms

a. A program/erase cycle is defined as switching the bits from 1->0 -> 1. Caution should be used when performing block erases, as repeated block erases can shorten the number of guaranteed erase cycles, see "Flash Memory Programming" on page 333.

26.1.6 GPIO Module Characteristics

Table 26-6. GPIO Module DC Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
R_{GPIOPU}	GPIO internal pull-up resistor	50	-	110	kΩ
R_{GPIOPD}	GPIO internal pull-down resistor	55	-	180	kΩ
I_{LKG}	GPIO input leakage current ^a	-	-	2	μA

a. The leakage current is measured with GND or V_{DD} applied to the corresponding pin(s). The leakage of digital port pins is measured individually. The port pin is configured as an input and the pullup/pulldown resistor is disabled.

26.1.7 USB Module Characteristics

The Stellaris® 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 LM3S9L97 microcontroller and specific to the Stellaris microcontroller design. An external component resistor is needed as specified in Table 26-7.

Table 26-7. USB Controller DC Characteristics

Parameter	Parameter Name	Value	Unit
R_{UBIAS}	Value of the pull-down resistor on the <code>USB0RBIAS</code> pin	$9.1K \pm 1\%$	Ω

26.1.8 Ethernet Controller Characteristics

Table 26-8. Ethernet Controller DC Characteristics

Parameter	Parameter Name	Value	Unit
R_{EBIAS}	Value of the pull-down resistor on the <code>ERBIAS</code> pin	$12.4K \pm 1\%$	Ω

26.1.9 Current Specifications

This section provides information on typical and maximum power consumption under various conditions.

26.1.9.1 Preliminary Current Consumption Specifications

The following table provides preliminary figures for current consumption while ongoing characterization is completed.

Table 26-9. Preliminary Current Consumption

Parameter	Parameter Name	Conditions	Nom	Max	Unit
I_{DD_RUN}	Run mode 1 (Flash loop)	$V_{DD} = 3.3V$ Code= <code>while(1){}</code> executed in Flash Peripherals = All ON System Clock = 50 MHz (with PLL) Temp = 25°C	80	-	mA
I_{DD_SLEEP}	Sleep mode	$V_{DD} = 3.3V$ Peripherals = All clock gated System Clock = 50 MHz (with PLL) Temp = 25°C	8	-	mA

Table 26-9. Preliminary Current Consumption (continued)

Parameter	Parameter Name	Conditions	Nom	Max	Unit
$I_{DD_DEEPSLEEP}$	Deep-sleep mode	Peripherals = All OFF System Clock = IOSC30KHZ/64 Temp = 25°C	550	-	µA
I_{HIB_NORTC}	Hibernate mode (external wake, RTC disabled, I/O not powered ^a)	$V_{BAT} = 3.0\text{ V}$ $V_{DD} = 0\text{ V}$ $V_{DDA} = 0\text{ V}$ Peripherals = All OFF System Clock = OFF Hibernate Module = 0 kHz	24	-	µA
I_{HIB_RTC}	Hibernate mode (RTC enabled, I/O not powered ^a)	$V_{BAT} = 3.0\text{ V}$ $V_{DD} = 0\text{ V}$ $V_{DDA} = 0\text{ V}$ Peripherals = All OFF System Clock = OFF Hibernate Module = 32 kHz	34	-	µA
I_{HIB_VDD3ON}	Hibernate mode (RTC enabled, I/O powered) ^b)	$V_{BAT} = 3.0\text{ V}$ $V_{DD} = 0\text{ V}$ $V_{DDA} = 0\text{ V}$ Peripherals = All OFF System Clock = OFF Hibernate Module = 32 kHz	pending ^c	-	µA

a. The VDD3ON mode must be disabled for the I/O ring to be unpowered.

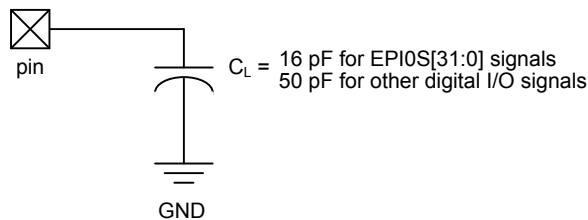
b. The VDD3ON mode must be enabled for the I/O ring to be powered.

c. Pending product characterization.

26.2 AC Characteristics

26.2.1 Load Conditions

Unless otherwise specified, the following conditions are true for all timing measurements.

Figure 26-1. 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 26-10. Phase Locked Loop (PLL) Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
f_{REF_XTAL}	Crystal reference ^a	3.579545	-	16.384	MHz
f_{REF_EXT}	External clock reference ^a	3.579545	-	16.384	MHz
f_{PLL}	PLL frequency ^b	-	400	-	MHz
T_{READY}	PLL lock time	0.562 ^c	-	1.38 ^d	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 26-11 on page 1236 shows the actual frequency of the PLL based on the crystal frequency used (defined by the XTAL field in the **RCC** register).

Table 26-11. 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	-
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 26-12. PIOSC Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$f_{PIOSC25}$	Internal 16-MHz precision oscillator frequency variance, factory calibrated at 25 °C	-	±0.25%	±1%	-
f_{PIOSCT}	Internal 16-MHz precision oscillator frequency variance, factory calibrated at 25 °C, across specified temperature range	-	-	±3%	-

Table 26-12. PIOSC Clock Characteristics (continued)

Parameter	Parameter Name	Min	Nom	Max	Unit
$f_{PIOSCUCAL}$	Internal 16-MHz precision oscillator frequency variance, user calibrated at a chosen temperature	-	$\pm 0.25\%$	$\pm 1\%$	-

26.2.2.3 Internal 30-kHz Oscillator Specifications

Table 26-13. 30-kHz Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$f_{IOSC30KHZ}$	Internal 30-KHz oscillator frequency	15	30	45	KHz

26.2.2.4 Hibernation Clock Source Specifications

Table 26-14. Hibernation Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
f_{HIBOSC}	Hibernation module oscillator frequency	-	4.194304	-	MHz
f_{HIBOSC_XTAL}	Crystal reference for hibernation oscillator	-	4.194304	-	MHz
f_{HIBOSC_EXT}	External clock reference for hibernation module	-	32.768	-	KHz
t_{HIBOSC_SETTLE}	Hibernation oscillator settling time ^a	-	-	10	ms

a. This parameter is highly sensitive to PCB layout and trace lengths, which may make this parameter time longer. Care must be taken in PCB design to minimize trace lengths and RLC (resistance, inductance, capacitance).

Table 26-15. HIB Oscillator Input Characteristics

Name	Value	Condition
Frequency	4.194304	MHz
Frequency tolerance	± 100	PPM
Oscillation mode	parallel	-
Equivalent series resistance (max)	200	Ω
Load capacitance, C_1 and C_2 ^a	12 - 22	pF
Drive level (typ)	100	μ w

a. Refer to the crystal manufacturer's recommended load capacitance.

26.2.2.5 Main Oscillator Specifications

Table 26-16. Main Oscillator Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
f_{MOSC}	Main oscillator frequency	1	-	16.384	MHz
t_{MOSC_PER}	Main oscillator period	61	-	1000	ns
t_{MOSC_SETTLE}	Main oscillator settling time	17.5	-	20	ms
$f_{REF_XTAL_BYPASS}$	Crystal reference using the main oscillator (PLL in BYPASS mode) ^a	1	-	16.384	MHz
$f_{REF_EXT_BYPASS}$	External clock reference (PLL in BYPASS mode) ^a	0	-	80	MHz

a. The ADC must be clocked from the PLL or directly from a 16-MHz clock source to operate properly.

Table 26-17. MOSC Oscillator Input Characteristics

Name	Value						Condition
Frequency	16	12	8	6	4	3.5	MHz
Frequency tolerance	± 100	PPM					
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

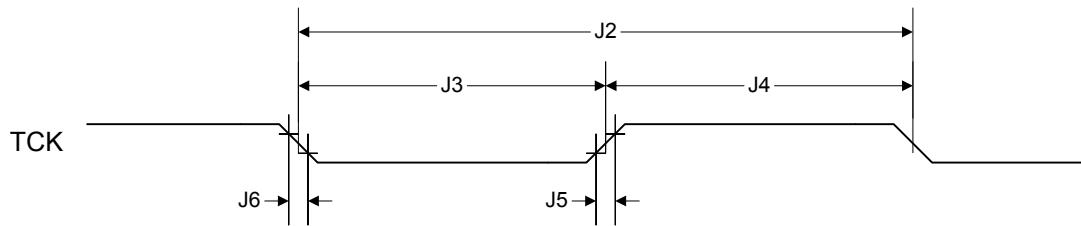
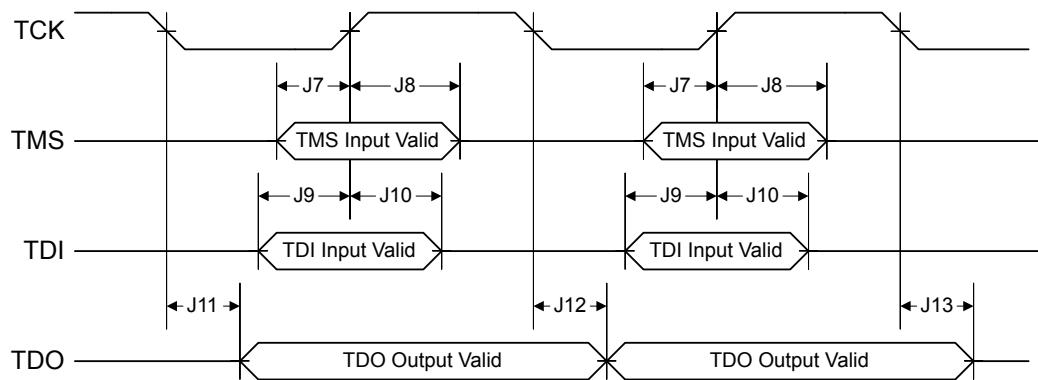
26.2.2.6 System Clock Specifications with ADC Operation**Table 26-18. System Clock Characteristics with ADC Operation**

Parameter	Parameter Name	Min	Nom	Max	Unit
f_{sysadc}	System clock frequency when the ADC module is operating (when PLL is bypassed)	16	-	-	MHz

26.2.3 JTAG and Boundary Scan**Table 26-19. JTAG Characteristics**

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J1	f_{TCK}	TCK operational clock frequency ^a	0	-	10	MHz
J2	t_{TCK}	TCK operational clock period	100	-	-	ns
J3	$t_{\text{TCK_LOW}}$	TCK clock Low time	-	t_{TCK}	-	ns
J4	$t_{\text{TCK_HIGH}}$	TCK clock High time	-	t_{TCK}	-	ns
J5	$t_{\text{TCK_R}}$	TCK rise time	0	-	10	ns
J6	$t_{\text{TCK_F}}$	TCK fall time	0	-	10	ns
J7	$t_{\text{TMS_SU}}$	TMS setup time to TCK rise	20	-	-	ns
J8	$t_{\text{TMS_HLD}}$	TMS hold time from TCK rise	20	-	-	ns
J9	$t_{\text{TDI_SU}}$	TDI setup time to TCK rise	25	-	-	ns
J10	$t_{\text{TDI_HLD}}$	TDI hold time from TCK rise	25	-	-	ns
J11 $t_{\text{TDO_ZDV}}$	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 $t_{\text{TDO_DV}}$	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 $t_{\text{TDO_DVZ}}$	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. A ratio of at least 8:1 must be kept between the system clock and TCK.

Figure 26-2. JTAG Test Clock Input Timing**Figure 26-3. JTAG Test Access Port (TAP) Timing**

26.2.4 Reset

Table 26-20. Reset Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
R1	V_{TH}	Power-On Reset threshold	-	2	-	V
R2	V_{BTH}	Brown-Out Reset threshold	2.85	2.9	2.95	V
R3	T_{POR}	Power-On Reset timeout	6	-	18	ms
R4	T_{BOR}	Brown-Out timeout	-	500	-	μ s
R5	T_{IRPOR}	Internal reset timeout after POR	-	-	2	ms
R6	T_{IRBOR}	Internal reset timeout after BOR	-	-	2	ms
R7	T_{IRHWR}	Internal reset timeout after hardware reset (\bar{RST} pin)	-	-	2	ms
R8	T_{IRSWR}	Internal reset timeout after software-initiated system reset	-	-	2	ms
R9	T_{IRWDR}	Internal reset timeout after watchdog reset	-	-	2	ms
R10	T_{IRMFR}	Internal reset timeout after MOSC failure reset	-	-	2	ms
R11	T_{MIN}	Minimum \bar{RST} pulse width	2	-	-	μ s
R12	$T_{VDDRISE}$	Supply voltage (V_{DD}) rise time (0V-3.0V)	-	-	10	ms
R13	T_{VDD2_3}	Supply voltage (V_{DD}) rise time (2.0V-3.0V)	-	-	6	ms

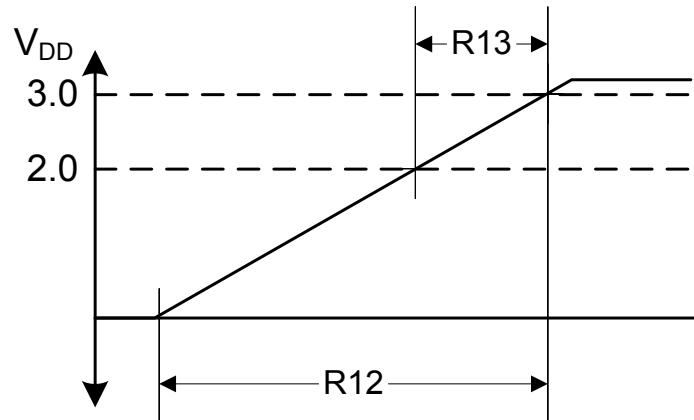
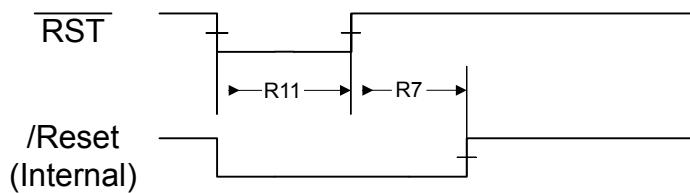
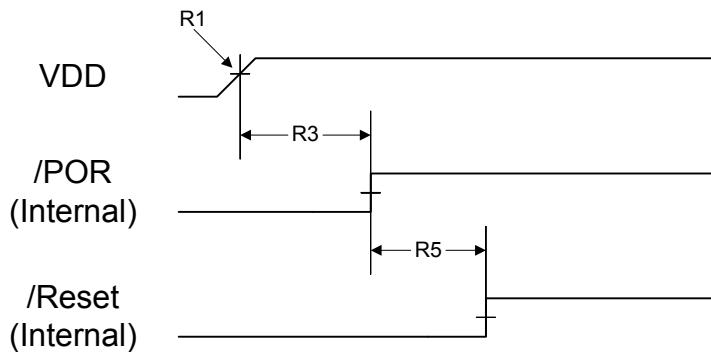
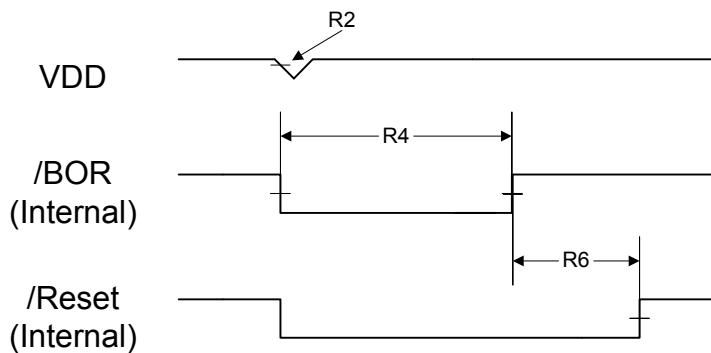
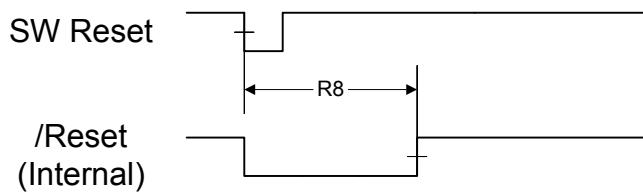
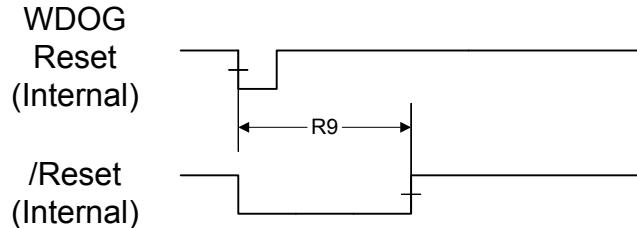
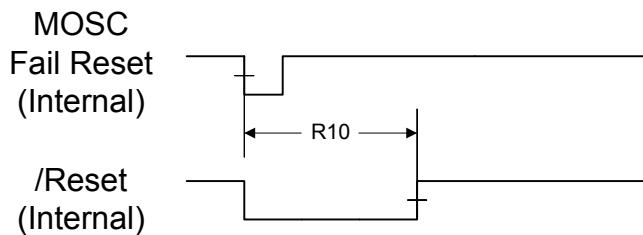
Figure 26-4. Power-On Reset and Voltage Parameters**Figure 26-5. External Reset Timing (\overline{RST})****Figure 26-6. Power-On Reset Timing****Figure 26-7. Brown-Out Reset Timing**

Figure 26-8. Software Reset Timing**Figure 26-9. Watchdog Reset Timing****Figure 26-10. MOSC Failure Reset Timing**

26.2.5 Sleep Modes

Table 26-21. Sleep Modes AC Characteristics^a

Parameter No	Parameter	Parameter Name	Min	Nom	Max	Unit
D1	t_{WAKE_S}	Time to wake from interrupt in sleep or deep-sleep mode, not using the PLL	-	-	7	system clocks
D2	$t_{WAKE_PLL_S}$	Time to wake from interrupt in sleep or deep-sleep mode when using the PLL	-	-	T_{READY}	ms
D3	t_{ENTER_DS}	Time to enter deep-sleep mode from sleep request	-	0	16^b	ms

a. Values in this table assume the IOSC is the clock source during sleep or deep-sleep mode.

b. Nominal specification occurs 99.9995% of the time.

26.2.6 Hibernation Module

The Hibernation Module requires special system implementation considerations because it is intended to power down all other sections of its host device, refer to "Hibernation Module" on page 301.

Table 26-22. Hibernation Module AC Characteristics

Parameter No	Parameter	Parameter Name	Min	Nom	Max	Unit
H1	t_{HIB_LOW}	Internal 32.768 KHz clock reference rising edge to \overline{HIB} asserted	20	-	-	μs
H2	t_{HIB_HIGH}	Internal 32.768 KHz clock reference rising edge to \overline{HIB} deasserted	-	30	-	μs
H3	$t_{WAKE_TO_HIB}$	WAKE assert to \overline{HIB} desassert (wake up time), internal Hibernation oscillator running during hibernation ^a	62	-	124	μs
H4	$t_{WAKE_TO_HIB}$	WAKE assert to \overline{HIB} desassert (wake up time), internal Hibernation oscillator stopped during hibernation ^a	-	-	10	ms
H5	t_{WAKE_CLOCK}	WAKE assertion time, internal Hibernation oscillator running during hibernation	62	-	-	μs
H6	$t_{WAKE_NOCLOCK}$	WAKE assertion time, internal Hibernation oscillator stopped during hibernation ^b	10	-	-	ms
H7	$t_{HIB_REG_ACCESS}$	Access time to or from a non-volatile register in HIB module to complete	92	-	-	μs
H8	$t_{HIB_TO_HIB}$	\overline{HIB} high time between assertions	100	-	-	ms
H9	t_{ENTER_HIB}	Time to enter hibernation mode from hibernation request	-	0	50 ^c	ms
H10	$t_{VDDRISE_HIB}$	Supply voltage (V_{DD}) rise time when waking from hibernation (1.8V-3.0V)	-	-	1.5	ms

a. Code begins executing after the time period specified by T_{IRPOR} following the deassertion of \overline{HIB} .

b. This mode is used when the PINWEN bit is set and the RTCEN bit is clear in the **HIBCTL** register.

c. Nominal specification occurs 99.998% of the time.

Figure 26-11. Hibernation Module Timing with Internal Oscillator Running in Hibernation

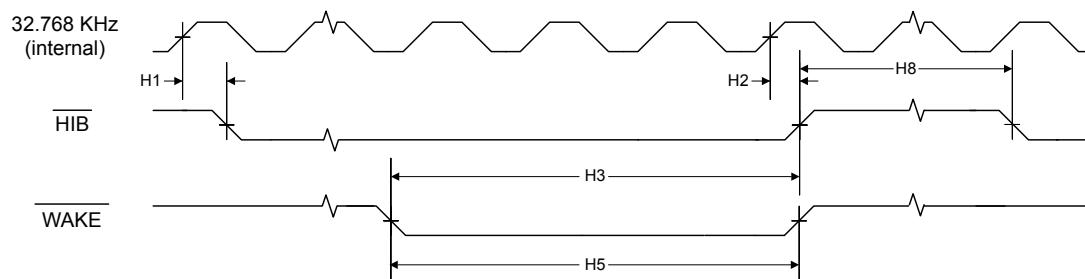


Figure 26-12. Hibernation Module Timing with Internal Oscillator Stopped in Hibernation

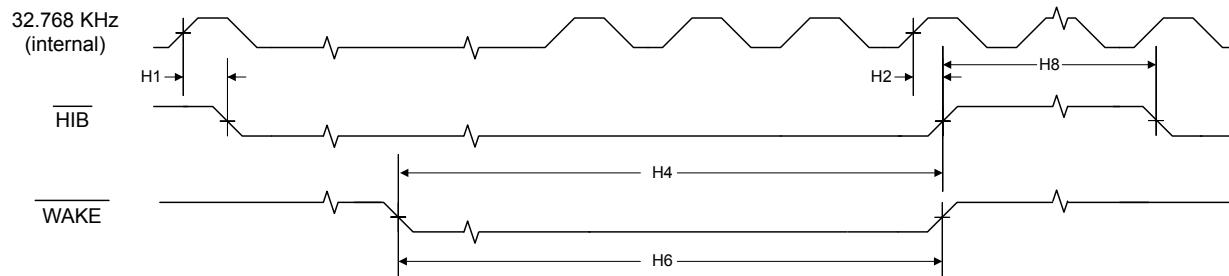
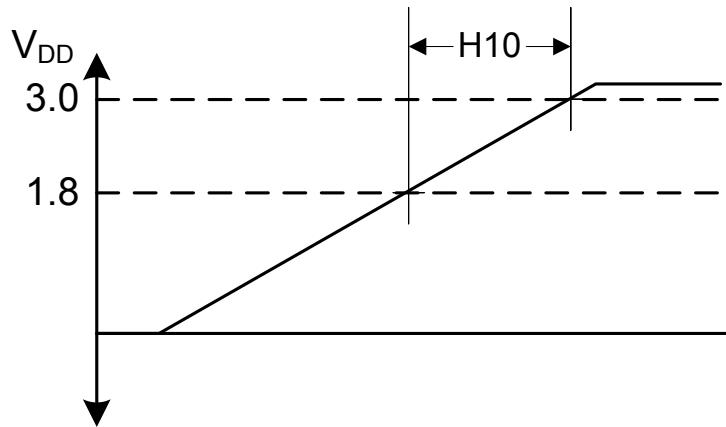


Figure 26-13. VDD Ramp when Waking from Hibernation

26.2.7 General-Purpose I/O (GPIO)

Note: All GPIOs are 5-V tolerant.

Table 26-23. GPIO Characteristics

Parameter	Parameter Name	Condition	Min	Nom	Max	Unit
t_{GPIOR}	GPIO Rise Time (from 20% to 80% of V_{DD})	2-mA drive	-	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_{GPIOF}	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

26.2.8 Analog-to-Digital Converter (ADC)

Table 26-24. ADC Characteristics^a

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{ADCIN}	Maximum single-ended, full-scale analog input voltage, using internal reference	-	-	3.0	V
	Maximum single-ended, full-scale analog input voltage, using external reference	-	-	V_{REFA}	V
	Minimum single-ended, full-scale analog input voltage	0.0	-	-	V
	Maximum differential, full-scale analog input voltage, using internal reference	-	-	1.5	V
	Maximum differential, full-scale analog input voltage, using external reference	-	-	$V_{\text{REFA}}/2$	V
	Minimum differential, full-scale analog input voltage	0.0	-	-	V
N	Resolution	10			bits
f_{ADC}	ADC internal clock frequency ^b	14	16	18	MHz

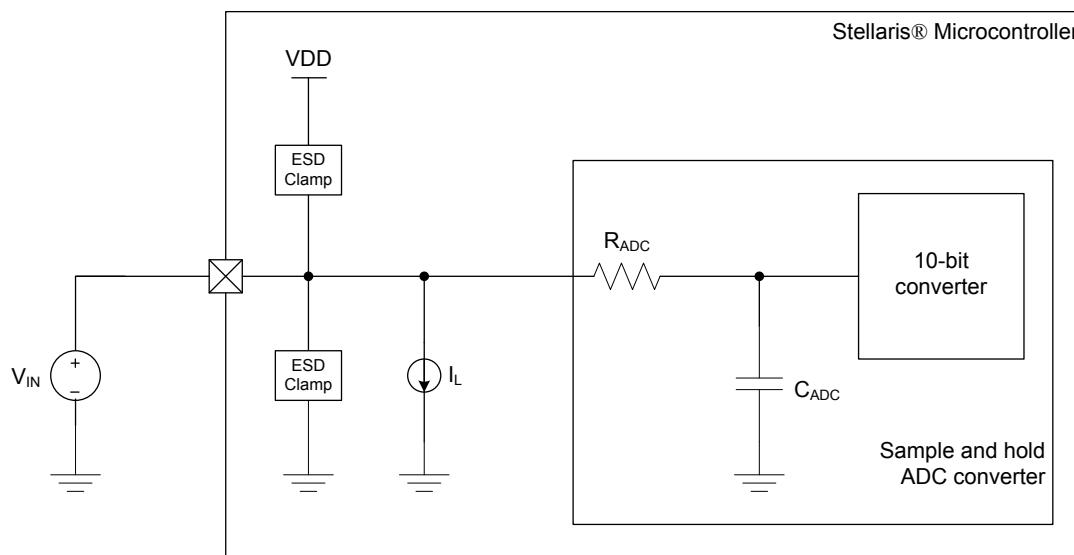
Table 26-24. ADC Characteristics (continued)

Parameter	Parameter Name	Min	Nom	Max	Unit
$t_{ADCCONV}$	Conversion time ^c		1		μs
$f_{ADCCONV}$	Conversion rate ^c		1000		k samples/s
t_{LT}	Latency from trigger to start of conversion	-	2	-	system clocks
I_L	ADC input leakage	-	-	± 1.0	μA
R_{ADC}	ADC equivalent resistance	-	-	10	kΩ
C_{ADC}	ADC equivalent capacitance	0.9	1.0	1.1	pF
E_L	Integral nonlinearity error	-	-	± 1	LSB
E_D	Differential nonlinearity error	-	-	± 1	LSB
E_O	Offset error	-	-	± 1	LSB
E_G	Full-scale gain error	-	-	± 3	LSB
E_{TS}	Temperature sensor accuracy	-	-	± 5	°C

a. The ADC reference voltage is 3.0 V. This reference voltage is internally generated from the 3.3 VDDA supply by a band gap circuit.

b. The ADC must be clocked from the PLL or directly from an external clock source to operate properly.

c. The conversion time and rate scale from the specified number if the ADC internal clock frequency is any value other than 16 MHz.

Figure 26-14. ADC Input Equivalency Diagram**Table 26-25. ADC Module External Reference Characteristics^a**

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{REFA}	External voltage reference for ADC ^b	2.4	-	3.06	V
I_L	External voltage reference leakage current	-	± 1.0	-	μA

a. Care must be taken to supply a reference voltage of acceptable quality.

b. Ground is always used as the reference level for the minimum conversion value.

Table 26-26. ADC Module Internal Reference Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{REFI}	Internal voltage reference for ADC	-	3.0	-	V
E_{IR}	Variation across temperature for a given device	-	-	± 2.5	%

26.2.9 Synchronous Serial Interface (SSI)**Table 26-27. 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 ^a	-	4	6	ns
S5	t_{DMD}	Data from master valid delay time	0	-	1	system clocks
S6	t_{DMS}	Data from master setup time	1	-	-	system clocks
S7	t_{DMH}	Data from master hold time	2	-	-	system clocks
S8	t_{DSS}	Data from slave setup time	1	-	-	system clocks
S9	t_{DSH}	Data from slave hold time	2	-	-	system clocks

a. Note that the delays shown are using 8-mA drive strength.

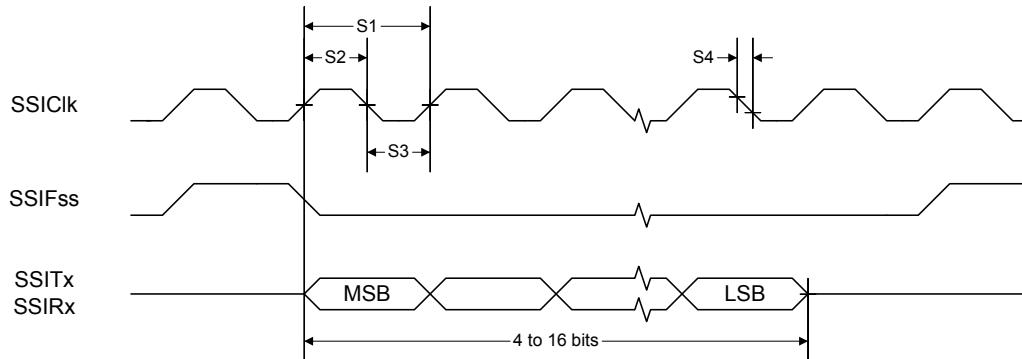
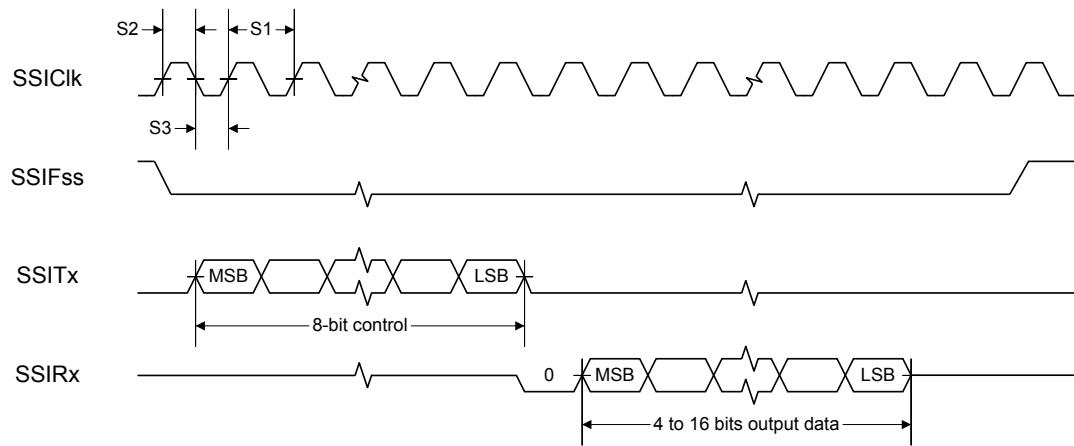
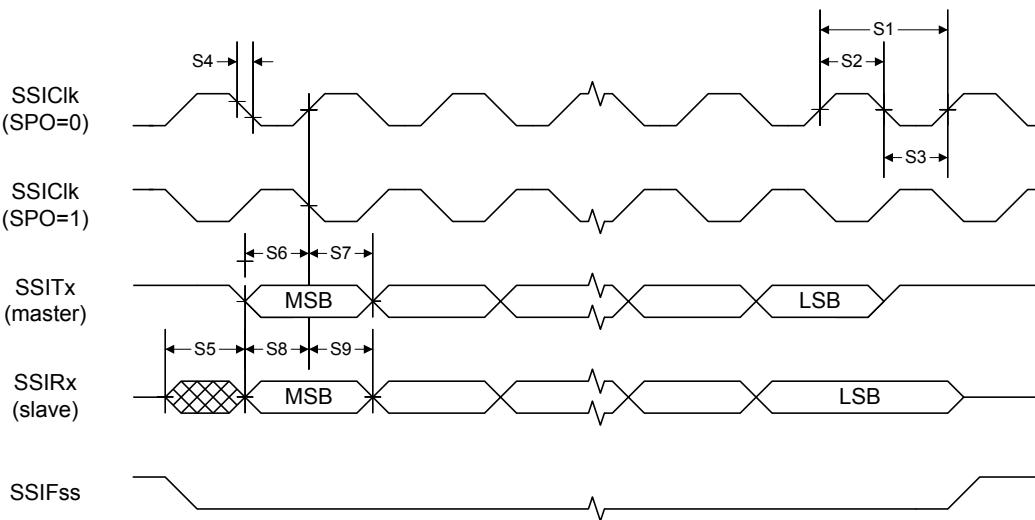
Figure 26-15. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement

Figure 26-16. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer**Figure 26-17. SSI Timing for SPI Frame Format (FRF=00), with SPH=1**

26.2.10 Inter-Integrated Circuit (I^2C) Interface

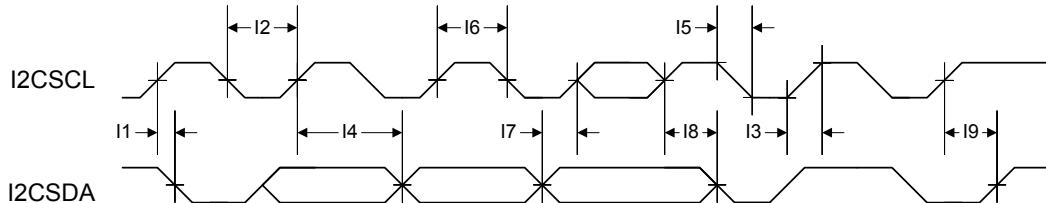
Table 26-28. I^2C Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
I1 ^a	t _{SCH}	Start condition hold time	36	-	-	system clocks
I2 ^a	t _{LP}	Clock Low period	36	-	-	system clocks
I3 ^b	t _{SRT}	I ₂ CSCL/I ₂ CSDA rise time ($V_{IL} = 0.5\text{ V}$ to $V_{IH} = 2.4\text{ V}$)	-	-	(see note b)	ns

Table 26-28. I²C Characteristics (continued)

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
I4 ^a	t _{DH}	Data hold time	2	-	-	system clocks
I5 ^c	t _{SFT}	I ² CSCL/I ² CSDA fall time ($V_{IH} = 2.4\text{ V}$ to $V_{IL} = 0.5\text{ V}$)	-	9	10	ns
I6 ^a	t _{HT}	Clock High time	24	-	-	system clocks
I7 ^a	t _{DS}	Data setup time	18	-	-	system clocks
I8 ^a	t _{SCSR}	Start condition setup time (for repeated start condition only)	36	-	-	system clocks
I9 ^a	t _{SCS}	Stop condition setup time	24	-	-	system clocks

- a. Values depend on the value programmed into the TPR bit in the **I²C Master Timer Period (I²CMTTPR)** register; a TPR programmed for the maximum I²CSCL frequency (TPR=0x2) results in a minimum output timing as shown in the table above. The I²C interface is designed to scale the actual data transition time to move it to the middle of the I²CSCL Low period. The actual position is affected by the value programmed into the TPR; however, the numbers given in the above values are minimum values.
- b. Because I²CSCL and I²CSDA are open-drain-type outputs, which the controller can only actively drive Low, the time I²CSCL or I²CSDA takes to reach a high level depends on external signal capacitance and pull-up resistor values.
- c. Specified at a nominal 50 pF load.

Figure 26-18. I²C Timing

26.2.11 Inter-Integrated Circuit Sound (I²S) Interface

Table 26-29. I²S Master Clock (Receive and Transmit)

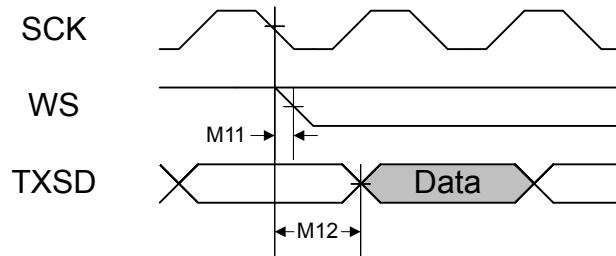
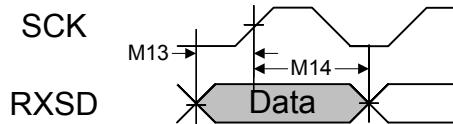
Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
M1	t _{MCLK_PER}	Cycle time	20.3	-	-	ns
M2	t _{MCLKRF}	Rise/fall time	See Table 26-23 on page 1243.			
M3	t _{MCLK_HIGH}	High time	10	-	-	ns
M4	t _{MCLK_LOW}	Low time	10	-	-	ns
M5	t _{MDC}	Duty cycle	48	-	52	%
M6	t _{MJITTER}	Jitter	-	-	2.5	ns

Table 26-30. I²S Slave Clock (Receive and Transmit)

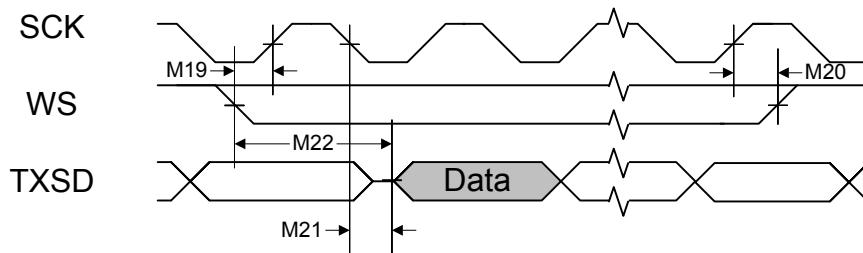
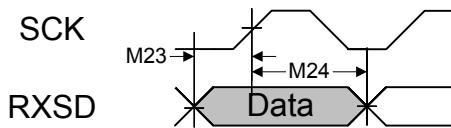
Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
M7	t _{SCLK_PER}	Cycle time	80	-	-	ns
M8	t _{SCLK_HIGH}	High time	40	-	-	ns
M9	t _{SCLK_LOW}	Low time	40	-	-	ns
M10	t _{SDC}	Duty cycle	-	50	-	%

Table 26-31. I²S Master Mode

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
M11	t_{MSWS}	SCK fall to WS valid	-	-	10	ns
M12	t_{MSD}	SCK fall to TXSD valid	-	-	10	ns
M13	t_{MSDS}	RXSD setup time to SCK rise	10	-	-	ns
M14	t_{MSDH}	RXSD hold time from SCK rise	10	-	-	ns

Figure 26-19. I²S Master Mode Transmit Timing**Figure 26-20. I²S Master Mode Receive Timing****Table 26-32. I²S Slave Mode**

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
M15	t_{SCLK_PER}	Cycle time	80	-	-	ns
M16	t_{SCLK_HIGH}	High time	40	-	-	ns
M17	t_{SCLK_LOW}	Low time	40	-	-	ns
M18	t_{SDC}	Duty cycle	-	50	-	%
M19	t_{SSETUP}	WS setup time to SCK rise	-	-	25	ns
M20	t_{SHOLD}	WS hold time from SCK rise	-	-	10	ns
M21	t_{SSD}	SCK fall to TXSD valid	-	-	20	ns
M22	t_{SLSD}	Left-justified mode, WS to TXSD	-	-	20	ns
M23	t_{SSDS}	RXSD setup time to SCK rise	10	-	-	ns
M24	t_{SSDH}	RXSD hold time from SCK rise	10	-	-	ns

Figure 26-21. I²S Slave Mode Transmit Timing**Figure 26-22. I²S Slave Mode Receive Timing**

26.2.12 Ethernet Controller

Table 26-33. 100BASE-TX Transmitter Characteristics^a

Parameter Name	Min	Nom	Max	Unit
Peak output amplitude	950	-	1050	mVpk
Output amplitude symmetry	98	-	102	%
Output overshoot	-	-	5	%
Rise/Fall time	3	-	5	ns
Rise/Fall time imbalance	-	-	500	ps
Duty cycle distortion	-	-	-	ps
Jitter	-	-	1.4	ns

a. Measured at the line side of the transformer.

Table 26-34. 100BASE-TX Transmitter Characteristics (informative)^a

Parameter Name	Min	Nom	Max	Unit
Return loss	16	-	-	dB
Open-circuit inductance	350	-	-	μH

a. The specifications in this table are included for information only. They are mainly a function of the external transformer and termination resistors used for measurements.

Table 26-35. 100BASE-TX Receiver Characteristics

Parameter Name	Min	Nom	Max	Unit
Signal detect assertion threshold	600	700	-	mVppd
Signal detect de-assertion threshold	350	425	-	mVppd
Differential input resistance	-	3.6	-	kΩ
Jitter tolerance (pk-pk)	4	-	-	ns
Baseline wander tracking	-80	-	+80	%

Table 26-35. 100BASE-TX Receiver Characteristics (continued)

Parameter Name	Min	Nom	Max	Unit
Signal detect assertion time	-	-	1000	μs
Signal detect de-assertion time	-	-	4	μs

Table 26-36. 10BASE-T Transmitter Characteristics^a

Parameter Name	Min	Nom	Max	Unit
Peak differential output signal	2.2	-	2.7	V
Harmonic content	27	-	-	dB
Link pulse width	-	100	-	ns
Start-of-idle pulse width	-	300 350	-	ns

a. The Manchester-encoded data pulses, the link pulse and the start-of-idle pulse are tested against the templates and using the procedures found in Clause 14 of *IEEE 802.3*.

Table 26-37. 10BASE-T Transmitter Characteristics (informative)^a

Parameter Name	Min	Nom	Max	Unit
Output return loss	15	-	-	dB
Output impedance balance	29-17log(f/10)	-	-	dB
Peak common-mode output voltage	-	-	50	mV
Common-mode rejection	-	-	100	mV
Common-mode rejection jitter	-	-	1	ns

a. The specifications in this table are included for information only. They are mainly a function of the external transformer and termination resistors used for measurements.

Table 26-38. 10BASE-T Receiver Characteristics

Parameter Name	Min	Nom	Max	Unit
Jitter tolerance (pk-pk)	30	26	-	ns
Input squelched threshold	340	440	540	mVppd
Differential input resistance	-	3.6	-	kΩ
Common-mode rejection	25	-	-	V

Table 26-39. Isolation Transformers^a

Name	Value	Condition
Turns ratio	1 CT : 1 CT	+/- 5%
Open-circuit inductance	350 uH (min)	@ 10 mV, 10 kHz
Leakage inductance	0.40 uH (max)	@ 1 MHz (min)
Inter-winding capacitance	25 pF (max)	
DC resistance	0.9 Ohm (max)	
Insertion loss	0.4 dB (typ)	0-65 MHz
HIPOT	1500	Vrms

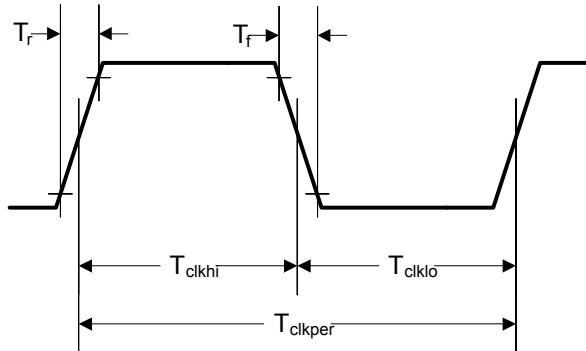
a. Two simple 1:1 isolation transformers are required at the line interface. Transformers with integrated common-mode chokes are recommended for exceeding FCC requirements. This table gives the recommended line transformer characteristics.

Note: The 100Base-TX amplitude specifications assume a transformer loss of 0.4 dB.

Table 26-40. Ethernet Reference Crystal

Name	Value	Condition
Frequency	25.00000	MHz
Frequency tolerance a	±50	PPM
Oscillation mode	Parallel resonance, fundamental mode	
Parameters at 25° C ±2° C; Drive level = 0.5 mW		
Drive level (typ)	50-100	µW
Shunt capacitance (max)	10	pF
Motional capacitance (min)	10	fF
Series resistance (max)	60	Ω
Spurious response (max)	> 5 dB below main within 500 kHz	

a. This tolerance provides a guard band for temperature stability and aging drift.

Figure 26-23. External XTLP Oscillator Characteristics**Table 26-41. External XTLP Oscillator Characteristics**

Parameter	Parameter Name	Min	Nom	Max	Unit
$XTLN_{ILV}$	XTLN Input Low Voltage	-	-	0.8	-
$XTLP_F$	XTLP Frequency ^a	-	25.0	-	-
T_{CLKPER}	XTLP Period ^a	-	40	-	-
$XTLP_{DC}$	XTLP Duty Cycle	40 40	-	60 60	%
T_r, T_f	Rise/Fall Time	-	-	4.0	ns
T_{JITTER}	Absolute Jitter	-	-	0.1	ns

a. IEEE 802.3 frequency tolerance ±50 ppm.

26.2.13 Universal Serial Bus (USB) Controller

The Stellaris 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*.

26.2.14 Analog Comparator

Table 26-42. Analog Comparator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{OS}	Input offset voltage	-	± 10	± 25	mV
V_{CM}	Input common mode voltage range	0	-	$V_{DD}-1.5$	V
C_{MRR}	Common mode rejection ratio	50	-	-	dB
T_{RT}	Response time	-	-	1	μs
T_{MC}	Comparator mode change to Output Valid	-	-	10	μs

Table 26-43. Analog Comparator Voltage Reference Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
R_{HR}	Resolution high range	-	$V_{DD}/31$	-	LSB
R_{LR}	Resolution low range	-	$V_{DD}/23$	-	LSB
A_{HR}	Absolute accuracy high range	-	-	$\pm 1/2$	LSB
A_{LR}	Absolute accuracy low range	-	-	$\pm 1/4$	LSB

A 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
The Cortex-M3 Processor															
R0, type R/W, , reset - (see page 80)															
DATA															
R1, type R/W, , reset - (see page 80)															
DATA															
R2, type R/W, , reset - (see page 80)															
DATA															
R3, type R/W, , reset - (see page 80)															
DATA															
R4, type R/W, , reset - (see page 80)															
DATA															
R5, type R/W, , reset - (see page 80)															
DATA															
R6, type R/W, , reset - (see page 80)															
DATA															
R7, type R/W, , reset - (see page 80)															
DATA															
R8, type R/W, , reset - (see page 80)															
DATA															
R9, type R/W, , reset - (see page 80)															
DATA															
R10, type R/W, , reset - (see page 80)															
DATA															
R11, type R/W, , reset - (see page 80)															
DATA															
R12, type R/W, , reset - (see page 80)															
DATA															
SP, type R/W, , reset - (see page 81)															
SP															
LR, type R/W, , reset 0xFFFF.FFFF (see page 82)															
LINK															
PC, type R/W, , reset - (see page 83)															
PC															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSR, type R/W, , reset 0x0100.0000 (see page 84)															
N	Z	C	V	Q	ICI / IT	THUMB									
		ICI / IT													ISRNUM
PRIMASK, type R/W, , reset 0x0000.0000 (see page 88)															
															PRIMASK
FAULTMASK, type R/W, , reset 0x0000.0000 (see page 89)															
															FAULTMASK
BASEPRI, type R/W, , reset 0x0000.0000 (see page 90)															
															BASEPRI
CONTROL, type R/W, , reset 0x0000.0000 (see page 91)															
															ASP TMPL
Cortex-M3 Peripherals															
System Timer (SysTick) Registers															
Base 0xE000.E000															
STCTRL, type R/W, offset 0x010, reset 0x0000.0004															
															COUNT
															CLK_SRC INTEN ENABLE
STRELOAD, type R/W, offset 0x014, reset 0x0000.0000															
															RELOAD
STCURRENT, type R/WC, offset 0x018, reset 0x0000.0000															
															CURRENT
Cortex-M3 Peripherals															
Nested Vectored Interrupt Controller (NVIC) Registers															
Base 0xE000.E000															
EN0, type R/W, offset 0x100, reset 0x0000.0000															
															INT
															INT
EN1, type R/W, offset 0x104, reset 0x0000.0000															
															INT
															INT
DIS0, type R/W, offset 0x180, reset 0x0000.0000															
															INT
															INT
DIS1, type R/W, offset 0x184, reset 0x0000.0000															
															INT
															INT
PEND0, type R/W, offset 0x200, reset 0x0000.0000															
															INT
															INT
PEND1, type R/W, offset 0x204, reset 0x0000.0000															
															INT
															INT
UNPEND0, type R/W, offset 0x280, reset 0x0000.0000															
															INT
															INT

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
UNPEND1, type R/W, offset 0x284, reset 0x0000.0000															
														INT	
								INT							
ACTIVE0, type RO, offset 0x300, reset 0x0000.0000															
								INT							
								INT							
ACTIVE1, type RO, offset 0x304, reset 0x0000.0000															
									INT						
									INT						
PRI0, type R/W, offset 0x400, reset 0x0000.0000															
INTD									INTC						
INTB									INTA						
PRI1, type R/W, offset 0x404, reset 0x0000.0000															
INTD									INTC						
INTB									INTA						
PRI2, type R/W, offset 0x408, reset 0x0000.0000															
INTD									INTC						
INTB									INTA						
PRI3, type R/W, offset 0x40C, reset 0x0000.0000															
INTD									INTC						
INTB									INTA						
PRI4, type R/W, offset 0x410, reset 0x0000.0000															
INTD									INTC						
INTB									INTA						
PRI5, type R/W, offset 0x414, reset 0x0000.0000															
INTD									INTC						
INTB									INTA						
PRI6, type R/W, offset 0x418, reset 0x0000.0000															
INTD									INTC						
INTB									INTA						
PRI7, type R/W, offset 0x41C, reset 0x0000.0000															
INTD									INTC						
INTB									INTA						
PRI8, type R/W, offset 0x420, reset 0x0000.0000															
INTD									INTC						
INTB									INTA						
PRI9, type R/W, offset 0x424, reset 0x0000.0000															
INTD									INTC						
INTB									INTA						
PRI10, type R/W, offset 0x428, reset 0x0000.0000															
INTD									INTC						
INTB									INTA						
PRI11, type R/W, offset 0x42C, reset 0x0000.0000															
INTD									INTC						
INTB									INTA						
PRI12, type R/W, offset 0x430, reset 0x0000.0000															
INTD									INTC						
INTB									INTA						
PRI13, type R/W, offset 0x434, reset 0x0000.0000															
INTD									INTC						
INTB									INTA						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
SWTRIG , type WO, offset 0xF00, reset 0x0000.0000																	
INTID																	
Cortex-M3 Peripherals																	
System Control Block (SCB) Registers																	
Base 0xE000.E000																	
ACTLR , type R/W, offset 0x008, reset 0x0000.0000																	
DISFOLD DISWBUF DISMCYC																	
CPUID , type RO, offset 0xD00, reset 0x412F.C230																	
IMP VAR CON																	
PARTNO REV																	
INTCTRL , type R/W, offset 0xD04, reset 0x0000.0000																	
NMISET		PENDSV	UNPENDSV	PENDSTSET	PENDSTCLR		ISRPRE	ISRPEND						VECPEND			
VECPEND		RETBASE												VECACT			
VTABLE , type R/W, offset 0xD08, reset 0x0000.0000																	
	BASE													OFFSET			
	OFFSET																
APINT , type R/W, offset 0xD0C, reset 0xFA05.0000																	
	VECTKEY																
ENDIANESS							PRIGROUP							SYSRESREQ	VECTCLRACT	VECTRESET	
SYSCTRL , type R/W, offset 0xD10, reset 0x0000.0000																	
														SEVONPEND	SLEEPDEEP	SLEEPEXIT	
CFGCTRL , type R/W, offset 0xD14, reset 0x0000.0200																	
							STKALIGN	BFHFNIGN						DIV0	UNALIGNED		
															MAINPEND	BASETHR	
SYSPRI1 , type R/W, offset 0xD18, reset 0x0000.0000																	
														USAGE			
	BUS													MEM			
SYSPRI2 , type R/W, offset 0xD1C, reset 0x0000.0000																	
	SVC																
SYSPRI3 , type R/W, offset 0xD20, reset 0x0000.0000																	
	TICK													PENDSV			
														DEBUG			
SYSHNDCTRL , type R/W, offset 0xD24, reset 0x0000.0000																	
SVC	BUSP	MEMP	USAGEP	TICK	PNDSV		MON	SVCA						USAGE	BUS	MEM	
BFARV			BSTKE	BUSTKE	IMPRE	PRECISE	IBUS	MMARV						USGA	BUSA	MEMA	
FAULTSTAT , type R/W1C, offset 0xD28, reset 0x0000.0000																	
							DIV0	UNALIGN						NOCP	INVPC	INVSTAT	UNDEF
							MSTKE							MUSTKE		DERR	IERR
HFAULTSTAT , type R/W1C, offset 0xD2C, reset 0x0000.0000																	
DBG	FORCED														VECT		
MMADDR , type R/W, offset 0xD34, reset -																	
														ADDR			
														ADDR			
FAULTADDR , type R/W, offset 0xD38, reset -																	
														ADDR			
														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																																
Cortex-M3 Peripherals																																															
Memory Protection Unit (MPU) Registers																																															
Base 0xE000.E000																																															
MPUTYPE, type RO, offset 0xD90, reset 0x0000.0800																																															
<table border="1" style="width: 100%; text-align: right;"> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>IREGION</td></tr> <tr> <td>DREGION</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>SEPARATE</td></tr> </table>																															IREGION	DREGION															SEPARATE
															IREGION																																
DREGION															SEPARATE																																
MPUCTRL, type R/W, offset 0xD94, reset 0x0000.0000																																															
<table border="1" style="width: 100%; text-align: right;"> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>PRIVDEFEN HFNMENA ENABLE</td></tr> </table>																															PRIVDEFEN HFNMENA ENABLE																
															PRIVDEFEN HFNMENA ENABLE																																
MPUNUMBER, type R/W, offset 0xD98, reset 0x0000.0000																																															
<table border="1" style="width: 100%; text-align: right;"> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>NUMBER</td></tr> </table>																															NUMBER																
															NUMBER																																
MPUBASE, type R/W, offset 0xD9C, reset 0x0000.0000																																															
<table border="1" style="width: 100%; text-align: right;"> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>ADDR</td></tr> <tr> <td>ADDR</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>VALID</td></tr> </table>																															ADDR	ADDR															VALID
															ADDR																																
ADDR															VALID																																
MPUBASE1, type R/W, offset 0xDA4, reset 0x0000.0000																																															
<table border="1" style="width: 100%; text-align: right;"> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>ADDR</td></tr> <tr> <td>ADDR</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>VALID</td></tr> </table>																															ADDR	ADDR															VALID
															ADDR																																
ADDR															VALID																																
MPUBASE2, type R/W, offset 0xDAC, reset 0x0000.0000																																															
<table border="1" style="width: 100%; text-align: right;"> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>ADDR</td></tr> <tr> <td>ADDR</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>VALID</td></tr> </table>																															ADDR	ADDR															VALID
															ADDR																																
ADDR															VALID																																
MPUBASE3, type R/W, offset 0xDB4, reset 0x0000.0000																																															
<table border="1" style="width: 100%; text-align: right;"> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>ADDR</td></tr> <tr> <td>ADDR</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>VALID</td></tr> </table>																															ADDR	ADDR															VALID
															ADDR																																
ADDR															VALID																																
MPUATTR, type R/W, offset 0xDA0, reset 0x0000.0000																																															
<table border="1" style="width: 100%; text-align: right;"> <tr> <td>XN</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>AP</td></tr> <tr> <td>SRD</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>TEX</td></tr> </table>																XN															AP	SRD															TEX
XN															AP																																
SRD															TEX																																
MPUATTR1, type R/W, offset 0xDA8, reset 0x0000.0000																																															
<table border="1" style="width: 100%; text-align: right;"> <tr> <td>XN</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>AP</td></tr> <tr> <td>SRD</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>TEX</td></tr> </table>																XN															AP	SRD															TEX
XN															AP																																
SRD															TEX																																
MPUATTR2, type R/W, offset 0xDB0, reset 0x0000.0000																																															
<table border="1" style="width: 100%; text-align: right;"> <tr> <td>XN</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>AP</td></tr> <tr> <td>SRD</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>TEX</td></tr> </table>																XN															AP	SRD															TEX
XN															AP																																
SRD															TEX																																
MPUATTR3, type R/W, offset 0xDB8, reset 0x0000.0000																																															
<table border="1" style="width: 100%; text-align: right;"> <tr> <td>XN</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>AP</td></tr> <tr> <td>SRD</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>TEX</td></tr> </table>																XN															AP	SRD															TEX
XN															AP																																
SRD															TEX																																
System Control																																															
Base 0x400F.E000																																															
DID0, type RO, offset 0x000, reset - (see page 215)																																															
<table border="1" style="width: 100%; text-align: right;"> <tr> <td>VER</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>CLASS</td></tr> <tr> <td>MAJOR</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>MINOR</td></tr> </table>																VER															CLASS	MAJOR															MINOR
VER															CLASS																																
MAJOR															MINOR																																
PBORCTL, type R/W, offset 0x030, reset 0x0000.7FFD (see page 217)																																															
<table border="1" style="width: 100%; text-align: right;"> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>BORIOR</td></tr> </table>																															BORIOR																
															BORIOR																																
RIS, type RO, offset 0x050, reset 0x0000.0000 (see page 218)																																															
<table border="1" style="width: 100%; text-align: right;"> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>MOSCPURIS USBPLLLRS PLLLRIS</td></tr> </table>																															MOSCPURIS USBPLLLRS PLLLRIS																
															MOSCPURIS USBPLLLRS PLLLRIS																																
IMC, type R/W, offset 0x054, reset 0x0000.0000 (see page 220)																																															
<table border="1" style="width: 100%; text-align: right;"> <tr> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>MOSCPURIM USBPLLLIM PLLLIM</td></tr> </table>																															MOSCPURIM USBPLLLIM PLLLIM																
															MOSCPURIM USBPLLLIM PLLLIM																																

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	
MISC, type R/W1C, offset 0x058, reset 0x0000.0000 (see page 222)																
								MOSCUPUMIS	USBPLLIMIS	PLLLMIS						
														BORMIS		
RESC, type R/W, offset 0x05C, reset - (see page 224)																
															MOSCFAIL	
									WDT1	SW	WDT0	BOR	POR	EXT		
RCC, type R/W, offset 0x060, reset 0x07E.3AD1 (see page 226)																
			ACG	SYS DIV	USE SYS DIV	USE PWM DIV	PWM DIV									
	PWRDN	BYPASS		XTAL		OSCSRC					IOSCDIS	MOSCDIS				
PLLCFG, type RO, offset 0x064, reset - (see page 231)																
														F	R	
GPIOHBCTL, type R/W, offset 0x06C, reset 0x0000.0000 (see page 232)																
								PORTJ	PORTH	PORTG	PORTF	PORTE	PORTD	PORTC	PORTB	PORTA
RCC2, type R/W, offset 0x070, reset 0x07C0.6810 (see page 234)																
USERCC2	DIV400	SYS DIV2			SYS DIV2LSB											
	USBPWRDN	PWRDN2	BYPASS2								OSCSRC2					
MOSCCTL, type R/W, offset 0x07C, reset 0x0000.0000 (see page 237)																
															CVAL	
DSLPCLKCFG, type R/W, offset 0x144, reset 0x0780.0000 (see page 238)																
		DSDIVORIDE														
														DSOSCSRC		
PIOSCCAL, type R/W, offset 0x150, reset 0x0000.0000 (see page 240)																
UTEN								CAL	UPDATE					UT		
PIOSCSTAT, type RO, offset 0x154, reset 0x0000.0040 (see page 242)																
														DT		
								RESULT						CT		
I2SMCLKCFG, type R/W, offset 0x170, reset 0x0000.0000 (see page 243)																
RXEN					RXI									RXF		
TXEN															TXF	
DID1, type RO, offset 0x004, reset - (see page 245)																
	VER		FAM											PARTNO		
	PINCOUNT													TEMP	PKG	
DC0, type RO, offset 0x008, reset 0x00BF.003F (see page 247)																
					SRAMSZ											
						FLASHSZ										
DC1, type RO, offset 0x010, reset - (see page 248)																
			WDT1		CAN1	CAN0					PWM			ADC1	ADC0	
	MINSYS DIV		MAXADC1SPD		MAXADC0SPD		MPU	HIB	TEMPSNS	PLL	WDT0	SWO	SWD	JTAG		
DC2, type RO, offset 0x014, reset 0x130F.5337 (see page 250)																
		I2S0		COMP1	COMP0									TIMER3	TIMER2	
	I2C1	I2C0		QEI1	QEI0						SSI1	SSI0		UART2	UART1	
DC3, type RO, offset 0x018, reset 0xBFFF.8FFF (see page 252)																
32KHZ	CCP5	CCP4	CCP3	CCP2	CCP1	CCP0	ADC0AIN7	ADC0AIN6	ADC0AIN5	ADC0AIN4	ADC0AIN3	ADC0AIN2	ADC0AIN1	ADC0AIN0		
PWMFAULT			C1O	C1PLUS	C1MINUS	C0O	C0PLUS	C0MINUS	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0		
DC4, type RO, offset 0x01C, reset 0x5104.F1FF (see page 254)																
	EPHY0	EMAC0			E1588									PICAL		
CCP7	CCP6	UDMA	ROM		GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE		GPIOD	GPIOC	GPIOB	GPIOA		

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
DC5, type RO, offset 0x020, reset 0x0F30.003F (see page 256)															
				PWMFAULT3	PWMFAULT2	PWMFAULT1	PWMFAULT0			PWM5	PWM4	PWM3	PWM2	PWM1	PWM0
DC6, type RO, offset 0x024, reset 0x0000.0013 (see page 258)															
										USB0PHY					USB0
DC7, type RO, offset 0x028, reset 0xFFFF.FFFF (see page 259)															
DMACH30	DMACH29	DMACH28	DMACH27	DMACH26	DMACH25	DMACH24	DMACH23	DMACH22	DMACH21	DMACH20	DMACH19	DMACH18	DMACH17	DMACH16	DMACH15
DMACH15	DMACH14	DMACH13	DMACH12	DMACH11	DMACH10	DMACH9	DMACH8	DMACH7	DMACH6	DMACH5	DMACH4	DMACH3	DMACH2	DMACH1	DMACH0
DC8, type RO, offset 0x02C, reset 0xFFFF.FFFF (see page 263)															
ADC1AIN15	ADC1AIN14	ADC1AIN13	ADC1AIN12	ADC1AIN11	ADC1AIN10	ADC1AIN9	ADC1AIN8	ADC1AIN7	ADC1AIN6	ADC1AIN5	ADC1AIN4	ADC1AIN3	ADC1AIN2	ADC1AIN1	ADC1AIN0
ADCOAIN15	ADCOAIN14	ADCOAIN13	ADCOAIN12	ADCOAIN11	ADCOAIN10	ADCOAIN9	ADCOAIN8	ADCOAIN7	ADCOAIN6	ADCOAIN5	ADCOAIN4	ADCOAIN3	ADCOAIN2	ADCOAIN1	ADCOAIN0
DC9, type RO, offset 0x190, reset 0x00FF.00FF (see page 265)															
								ADC1DC7	ADC1DC6	ADC1DC5	ADC1DC4	ADC1DC3	ADC1DC2	ADC1DC1	ADC1DC0
								ADC0DC7	ADC0DC6	ADC0DC5	ADC0DC4	ADC0DC3	ADC0DC2	ADC0DC1	ADC0DC0
NVMSTAT, type RO, offset 0x1A0, reset 0x0000.0001 (see page 267)															
															FWB
RCGC0, type R/W, offset 0x100, reset 0x00000040 (see page 268)															
		WDT1		CAN1	CANO				PWM			ADC1	ADC0		
			MAXADC1SPD	MAXADC0SPD		HIB			WDT0						
SCGC0, type R/W, offset 0x110, reset 0x00000040 (see page 271)															
		WDT1		CAN1	CANO				PWM			ADC1	ADC0		
			MAXADC1SPD	MAXADC0SPD		HIB			WDT0						
DCGC0, type R/W, offset 0x120, reset 0x00000040 (see page 274)															
		WDT1		CAN1	CANO				PWM			ADC1	ADC0		
				HIB					WDT0						
RCGC1, type R/W, offset 0x104, reset 0x00000000 (see page 276)															
		I2S0		COMP1	COMP0				TIMER3	TIMER2	TIMER1	TIMER0			
I2C1	I2C0		QEI1	QEIO			SSI1	SSI0		UART2	UART1	UART0			
SCGC1, type R/W, offset 0x114, reset 0x00000000 (see page 279)															
		I2S0		COMP1	COMP0				TIMER3	TIMER2	TIMER1	TIMER0			
I2C1	I2C0		QEI1	QEIO			SSI1	SSI0		UART2	UART1	UART0			
DCGC1, type R/W, offset 0x124, reset 0x00000000 (see page 282)															
		I2S0		COMP1	COMP0				TIMER3	TIMER2	TIMER1	TIMER0			
I2C1	I2C0		QEI1	QEIO			SSI1	SSI0		UART2	UART1	UART0			
RCGC2, type R/W, offset 0x108, reset 0x00000000 (see page 285)															
	EPHY0	EMAC0													USB0
		UDMA					GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
SCGC2, type R/W, offset 0x118, reset 0x00000000 (see page 288)															
	EPHY0	EMAC0													USB0
		UDMA					GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
DCGC2, type R/W, offset 0x128, reset 0x00000000 (see page 291)															
	EPHY0	EMAC0													USB0
		UDMA					GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
SRCR0, type R/W, offset 0x040, reset 0x00000000 (see page 294)															
		WDT1		CAN1	CANO				PWM			ADC1	ADC0		
				HIB					WDT0						
SRCR1, type R/W, offset 0x044, reset 0x00000000 (see page 296)															
		I2S0		COMP1	COMP0				TIMER3	TIMER2	TIMER1	TIMER0			
I2C1	I2C0		QEI1	QEIO			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	
SRCR2, type R/W, offset 0x048, reset 0x00000000 (see page 299)																
	EPHY0		EMAC0												USB0	
		UDMA					GPIOJ	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	
Hibernation Module																
Base 0x400F.C000																
HIBRTC0, type RO, offset 0x000, reset 0x0000.0000 (see page 312)																
								RTCC								
								RTCC								
HIBRTCM0, type R/W, offset 0x004, reset 0xFFFF.FFFF (see page 313)																
								RTCM0								
								RTCM0								
HIBRTCM1, type R/W, offset 0x008, reset 0xFFFF.FFFF (see page 314)																
								RTCM1								
								RTCM1								
HIBRTCLD, type R/W, offset 0x00C, reset 0xFFFF.FFFF (see page 315)																
								RTCLD								
								RTCLD								
HIBCTL, type R/W, offset 0x010, reset 0x8000.0000 (see page 316)																
WRC								VDD3ON	VABORT	CLK32EN	LOWBATEN	PINWEN	RTCWEN	CLKSEL	HIBREQ	RTCEN
HIBIM, type R/W, offset 0x014, reset 0x0000.0000 (see page 319)																
													EXTW	LOWBAT	RTCALT1	RTCALT0
HIBRIS, type RO, offset 0x018, reset 0x0000.0000 (see page 321)																
													EXTW	LOWBAT	RTCALT1	RTCALT0
HIBMIS, type RO, offset 0x01C, reset 0x0000.0000 (see page 323)																
													EXTW	LOWBAT	RTCALT1	RTCALT0
HIBIC, type R/W1C, offset 0x020, reset 0x0000.0000 (see page 325)																
													EXTW	LOWBAT	RTCALT1	RTCALT0
HIBRTCT, type R/W, offset 0x024, reset 0x0000.7FFF (see page 326)																
													TRIM			
HIBDATA, type R/W, offset 0x030-0x12C, reset - (see page 327)																
													RTD			
													RTD			
Internal Memory																
Flash Memory Registers (Flash Control Offset)																
Base 0x400F.D000																
FMA, type R/W, offset 0x000, reset 0x0000.0000																
															OFFSET	
													OFFSET			
FMD, type R/W, offset 0x004, reset 0x0000.0000																
													DATA			
													DATA			
FMC, type R/W, offset 0x008, reset 0x0000.0000																
													WRKEY			
													COMT	MERASE	ERASE	WRITE

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
FCRIS, type RO, offset 0x00C, reset 0x0000.0000																
														PRIS	ARIS	
FCIM, type R/W, offset 0x010, reset 0x0000.0000																
														PMASK	AMASK	
FCMISC, type R/W1C, offset 0x014, reset 0x0000.0000																
														PMISC	AMISC	
FMC2, type R/W, offset 0x020, reset 0x0000.0000																
														WRKEY	WRBUF	
FWBVAL, type R/W, offset 0x030, reset 0x0000.0000																
														FWB[n]	FWB[n]	
FCTL, type R/W, offset 0x0F8, reset 0x0000.0000																
														USDACK	USDREQ	
FWBn, type R/W, offset 0x100 - 0x17C, reset 0x0000.0000																
														DATA	DATA	
Internal Memory																
Memory Registers (System Control Offset)																
Base 0x400F.E000																
RMCTL, type R/W1C, offset 0x0F0, reset -																
															BA	
FMPRE0, type R/W, offset 0x130 and 0x200, reset 0xFFFF.FFFF																
														READ_ENABLE	READ_ENABLE	
FMPPE0, type R/W, offset 0x134 and 0x400, reset 0xFFFF.FFFF																
														PROG_ENABLE	PROG_ENABLE	
BOOTCFG, type R/W, offset 0x1D0, reset 0xFFFF.FFFE																
NW																
	PORT		PIN		POL	EN									DBG1	DBG0
USER_REG0, type R/W, offset 0x1E0, reset 0xFFFF.FFFF																
NW														DATA	DATA	
USER_REG1, type R/W, offset 0x1E4, reset 0xFFFF.FFFF																
NW														DATA	DATA	
USER_REG2, type R/W, offset 0x1E8, reset 0xFFFF.FFFF																
NW														DATA	DATA	
USER_REG3, type R/W, offset 0x1EC, reset 0xFFFF.FFFF																
NW														DATA	DATA	
FMPRE1, type R/W, offset 0x204, reset 0xFFFF.FFFF																
														READ_ENABLE	READ_ENABLE	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMPRE2, type R/W, offset 0x208, reset 0x0000.0000															
								READ_ENABLE							
								READ_ENABLE							
FMPRE3, type R/W, offset 0x20C, reset 0x0000.0000															
								READ_ENABLE							
								READ_ENABLE							
FMPPE1, type R/W, offset 0x404, reset 0xFFFF.FFFF															
								PROG_ENABLE							
								PROG_ENABLE							
FMPPE2, type R/W, offset 0x408, reset 0x0000.0000															
								PROG_ENABLE							
								PROG_ENABLE							
FMPPE3, type R/W, offset 0x40C, reset 0x0000.0000															
								PROG_ENABLE							
								PROG_ENABLE							
Micro Direct Memory Access (μDMA)															
μDMA Channel Control Structure (Offset from Channel Control Table Base)															
Base n/a															
DMASRCENDP, type R/W, offset 0x000, reset -															
								ADDR							
								ADDR							
DMADSTENDP, type R/W, offset 0x004, reset -															
								ADDR							
								ADDR							
DMACHCTL, type R/W, offset 0x008, reset -															
DSTINC	DSTSIZE	SRCINC	SRCSIZE												ARBSIZE
ARBSIZE			XFERSIZE									NXTUSEBURST		XFERMODE	
Micro Direct Memory Access (μDMA)															
μDMA Registers (Offset from μDMA Base Address)															
Base 0x400F.F000															
DMASTAT, type RO, offset 0x000, reset 0x001F.0000															
														DMACHANS	
														STATE	
															MASTEN
DMACFG, type WO, offset 0x004, reset -															
															MASTEN
DMACTLBASE, type R/W, offset 0x008, reset 0x0000.0000															
								ADDR							
								ADDR							
DMAALTBASE, type RO, offset 0x00C, reset 0x0000.0200															
								ADDR							
								ADDR							
DMAWAITSTAT, type RO, offset 0x010, reset 0x0000.0000															
								WAITREQ[n]							
								WAITREQ[n]							
DMASWREQ, type WO, offset 0x014, reset -															
								SWREQ[n]							
								SWREQ[n]							
DMAUSEBURSTSET, type R/W, offset 0x018, reset 0x0000.0000															
								SET[n]							
								SET[n]							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAUSEBURSTCLR , type WO, offset 0x01C, reset -															
								CLR[n]							
								CLR[n]							
DMAREQMASKSET , type R/W, offset 0x020, reset 0x0000.0000															
								SET[n]							
								SET[n]							
DMAREQMASKCLR , type WO, offset 0x024, reset -															
								CLR[n]							
								CLR[n]							
DMAENASET , type R/W, offset 0x028, reset 0x0000.0000															
								SET[n]							
								SET[n]							
DMAENACLR , type WO, offset 0x02C, reset -															
								CLR[n]							
								CLR[n]							
DMAALTSET , type R/W, offset 0x030, reset 0x0000.0000															
								SET[n]							
								SET[n]							
DMAALTCLR , type WO, offset 0x034, reset -															
								CLR[n]							
								CLR[n]							
DMAPRIOSET , type R/W, offset 0x038, reset 0x0000.0000															
								SET[n]							
								SET[n]							
DMAPRIOCLR , type WO, offset 0x03C, reset -															
								CLR[n]							
								CLR[n]							
DMAERRCLR , type R/W, offset 0x04C, reset 0x0000.0000															
															ERRCLR
DMACHASGN , type R/W, offset 0x500, reset 0x0000.0000															
								CHASGN[n]							
								CHASGN[n]							
DMAPeriphID0 , type RO, offset 0xFE0, reset 0x0000.0030															
															PID0
DMAPeriphID1 , type RO, offset 0xFE4, reset 0x0000.00B2															
															PID1
DMAPeriphID2 , type RO, offset 0xFE8, reset 0x0000.000B															
															PID2
DMAPeriphID3 , type RO, offset 0xFEC, reset 0x0000.0000															
															PID3
DMAPeriphID4 , type RO, offset 0xFD0, reset 0x0000.0004															
															PID4
DAPCellID0 , type RO, offset 0xFF0, reset 0x0000.000D															
															CID0

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
DMACellID1, type RO, offset 0xFF4, reset 0x0000.00F0															
CID1															
DMACellID2, type RO, offset 0xFF8, reset 0x0000.0005															
CID2															
DMACellID3, type RO, offset 0xFFC, reset 0x0000.00B1															
CID3															
General-Purpose Input/Outputs (GPIOs)															
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															
GPIODATA, type R/W, offset 0x000, reset 0x0000.0000 (see page 437)															
DATA															
GPIODIR, type R/W, offset 0x400, reset 0x0000.0000 (see page 438)															
DIR															
GPIOIS, type R/W, offset 0x404, reset 0x0000.0000 (see page 439)															
IS															
GPIOIBE, type R/W, offset 0x408, reset 0x0000.0000 (see page 440)															
IBE															
GPIOIEV, type R/W, offset 0x40C, reset 0x0000.0000 (see page 441)															
IEV															
GPIOIM, type R/W, offset 0x410, reset 0x0000.0000 (see page 442)															
IME															
GPIORIS, type RO, offset 0x414, reset 0x0000.0000 (see page 443)															
RIS															
GPIOOMIS, type RO, offset 0x418, reset 0x0000.0000 (see page 444)															
MIS															
GPIOICR, type W1C, offset 0x41C, reset 0x0000.0000 (see page 446)															
IC															
GPIOAFSEL, type R/W, offset 0x420, reset - (see page 447)															
AFSEL															

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												
GPIODR2R , type R/W, offset 0x500, reset 0x0000.00FF (see page 449)																											
DRV2																											
GPIODR4R , type R/W, offset 0x504, reset 0x0000.0000 (see page 450)																											
DRV4																											
GPIODR8R , type R/W, offset 0x508, reset 0x0000.0000 (see page 451)																											
DRV8																											
GPIOODR , type R/W, offset 0x50C, reset 0x0000.0000 (see page 452)																											
ODE																											
GPIOPUR , type R/W, offset 0x510, reset - (see page 453)																											
PUE																											
GPIOPDR , type R/W, offset 0x514, reset 0x0000.0000 (see page 455)																											
PDE																											
GPIOSLR , type R/W, offset 0x518, reset 0x0000.0000 (see page 457)																											
SRL																											
GPIODEN , type R/W, offset 0x51C, reset - (see page 458)																											
DEN																											
GPIOLOCK , type R/W, offset 0x520, reset 0x0000.0001 (see page 460)																											
LOCK																											
GPIOCR , type -, offset 0x524, reset - (see page 461)																											
CR																											
GPIOAMSEL , type R/W, offset 0x528, reset 0x0000.0000 (see page 463)																											
GPIOAMSEL																											
GPIOPCTL , type R/W, offset 0x52C, reset - (see page 465)																											
PMC7				PMC6				PMC5				PMC4															
PMC3				PMC2				PMC1				PMC0															
GPIOPeriphID4 , type RO, offset 0xFD0, reset 0x0000.0000 (see page 467)																											
PID4																											
GPIOPeriphID5 , type RO, offset 0xFD4, reset 0x0000.0000 (see page 468)																											
PID5																											
GPIOPeriphID6 , type RO, offset 0xFD8, reset 0x0000.0000 (see page 469)																											
PID6																											
GPIOPeriphID7 , type RO, offset 0xFDC, reset 0x0000.0000 (see page 470)																											
PID7																											
GPIOPeriphID0 , type RO, offset 0xFE0, reset 0x0000.0061 (see page 471)																											
PID0																											

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
GPIOPeriphID1 , type RO, offset 0xFE4, reset 0x0000.0000 (see page 472)															
PID1															
GPIOPeriphID2 , type RO, offset 0xFE8, reset 0x0000.0018 (see page 473)															
PID2															
GPIOPeriphID3 , type RO, offset 0xFEC, reset 0x0000.0001 (see page 474)															
PID3															
GPIOCellID0 , type RO, offset 0xFF0, reset 0x0000.000D (see page 475)															
CID0															
GPIOCellID1 , type RO, offset 0xFF4, reset 0x0000.00F0 (see page 476)															
CID1															
GPIOCellID2 , type RO, offset 0xFF8, reset 0x0000.0005 (see page 477)															
CID2															
GPIOCellID3 , type RO, offset 0xFFC, reset 0x0000.00B1 (see page 478)															
CID3															
General-Purpose Timers															
Timer 0 base: 0x4003.0000															
Timer 1 base: 0x4003.1000															
Timer 2 base: 0x4003.2000															
Timer 3 base: 0x4003.3000															
GPTMCFG , type R/W, offset 0x000, reset 0x0000.0000 (see page 495)															
GPTMCFG															
GPTMTAMR , type R/W, offset 0x004, reset 0x0000.0000 (see page 496)															
TASNAPS TAWOT TAMIE TACDIR TAAMS TACMR TAMR															
GPTMTBMR , type R/W, offset 0x008, reset 0x0000.0000 (see page 498)															
TBSNAPS TBWOT TBMIE TBCDIR TBAMS TBCMCR TBMR															
GPTMCTL , type R/W, offset 0x00C, reset 0x0000.0000 (see page 500)															
TBPWML TBOTE TBEVENT TBSTALL TBEN TAPWML TAOTE RTCEN TAEVENT TASTALL TAEN															
GPTMIMR , type R/W, offset 0x018, reset 0x0000.0000 (see page 503)															
TBMIM CBEIM CBMIM TBTOIM TAMIM RTCIM CAEIM CAMIM TATOIM															
GPTMRIS , type RO, offset 0x01C, reset 0x0000.0000 (see page 505)															
TBMRIS CBERIS CBMRIS TBTORIS TAMRIS RTCRIS CAERIS CAMRIS TATORIS															
GPTMMIS , type RO, offset 0x020, reset 0x0000.0000 (see page 508)															
TBMMIS CBEMIS CBMMIS TBTOVIS TAMMIS RTCMIS CAEMIS CAMMIS TATOMIS															
GPTMICR , type W1C, offset 0x024, reset 0x0000.0000 (see page 511)															
TBMCINT CBECINT CBMCINT TBTOCINT TAMCINT RTCCINT CAECINT CAMCINT TATOCINT															
GPTMTAILR , type R/W, offset 0x028, reset 0xFFFF.FFFF (see page 513)															
TAILR															

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
GPTMTBILR , type R/W, offset 0x02C, reset 0x0000.FFFF (see page 514)															
TBILR															
TBILR															
GPTMTAMATCHR , type R/W, offset 0x030, reset 0xFFFF.FFFF (see page 515)															
TAMR															
TAMR															
GPTMTBMATCHR , type R/W, offset 0x034, reset 0x0000.FFFF (see page 516)															
TBMR															
TBMR															
GPTMTAPR , type R/W, offset 0x038, reset 0x0000.0000 (see page 517)															
TAPSR															
GPTMTBPR , type R/W, offset 0x03C, reset 0x0000.0000 (see page 518)															
TBPSR															
GPTMTAPMR , type R/W, offset 0x040, reset 0x0000.0000 (see page 519)															
TAPSMR															
GPTMTBPMR , type R/W, offset 0x044, reset 0x0000.0000 (see page 520)															
TBPSMR															
GPTMTAR , type RO, offset 0x048, reset 0xFFFF.FFFF (see page 521)															
TAR															
GPTMTBR , type RO, offset 0x04C, reset 0x0000.FFFF (see page 522)															
TBR															
GPTMTAV , type RW, offset 0x050, reset 0xFFFF.FFFF (see page 523)															
TAV															
GPTMTBV , type RW, offset 0x054, reset 0x0000.FFFF (see page 524)															
TBV															
Watchdog Timers															
WDT0 base: 0x4000.0000															
WDT1 base: 0x4000.1000															
WDTLOAD , type R/W, offset 0x000, reset 0xFFFF.FFFF (see page 529)															
WDTLOAD															
WDTVALUE , type RO, offset 0x004, reset 0xFFFF.FFFF (see page 530)															
WDTVALUE															
WDTCTL , type R/W, offset 0x008, reset 0x0000.0000 (WDT0) and 0x8000.0000 (WDT1) (see page 531)															
WRC														RESEN	INTEN
WDTICR , type WO, offset 0x00C, reset - (see page 533)															
WDTINTCLR															
WDTRIS , type RO, offset 0x010, reset 0x0000.0000 (see page 534)															WDTRIS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDTMIS , type RO, offset 0x014, reset 0x0000.0000 (see page 535)															
															WDTMIS
WDTTEST , type R/W, offset 0x418, reset 0x0000.0000 (see page 536)															
							STALL								
WDTLOCK , type R/W, offset 0xC00, reset 0x0000.0000 (see page 537)															
								WDTLOCK							
WDTPeriphID4 , type RO, offset 0xFD0, reset 0x0000.0000 (see page 538)															
															PID4
WDTPeriphID5 , type RO, offset 0xFD4, reset 0x0000.0000 (see page 539)															
															PID5
WDTPeriphID6 , type RO, offset 0xFD8, reset 0x0000.0000 (see page 540)															
															PID6
WDTPeriphID7 , type RO, offset 0xFDC, reset 0x0000.0000 (see page 541)															
															PID7
WDTPeriphID0 , type RO, offset 0xFE0, reset 0x0000.0005 (see page 542)															
															PID0
WDTPeriphID1 , type RO, offset 0xFE4, reset 0x0000.0018 (see page 543)															
															PID1
WDTPeriphID2 , type RO, offset 0xFE8, reset 0x0000.0018 (see page 544)															
															PID2
WDTPeriphID3 , type RO, offset 0xFEC, reset 0x0000.0001 (see page 545)															
															PID3
WDTPCellID0 , type RO, offset 0xFF0, reset 0x0000.000D (see page 546)															
															CID0
WDTPCellID1 , type RO, offset 0xFF4, reset 0x0000.00F0 (see page 547)															
															CID1
WDTPCellID2 , type RO, offset 0xFF8, reset 0x0000.0006 (see page 548)															
															CID2
WDTPCellID3 , type RO, offset 0xFFC, reset 0x0000.00B1 (see page 549)															
															CID3
Analog-to-Digital Converter (ADC)															
ADC0 base: 0x4003.8000															
ADC1 base: 0x4003.9000															
ADCACTSS , type R/W, offset 0x000, reset 0x0000.0000 (see page 571)															
															ASEN3 ASEN2 ASEN1 ASEN0

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		
ADCRIS , type RO, offset 0x004, reset 0x0000.0000 (see page 572)																	
															INRDC		
														INR3	INR2	INR1	INR0
ADCIM , type R/W, offset 0x008, reset 0x0000.0000 (see page 574)																	
												DCONSS3	DCONSS2	DCONSS1	DCONSS0		
												MASK3	MASK2	MASK1	MASK0		
ADCISC , type R/W1C, offset 0x00C, reset 0x0000.0000 (see page 576)																	
												DCINSS3	DCINSS2	DCINSS1	DCINSS0		
												IN3	IN2	IN1	IN0		
ADCOSTAT , type R/W1C, offset 0x010, reset 0x0000.0000 (see page 579)																	
												OV3	OV2	OV1	OV0		
ADCEMUX , type R/W, offset 0x014, reset 0x0000.0000 (see page 581)																	
												EM3	EM2	EM1	EM0		
ADCUSTAT , type R/W1C, offset 0x018, reset 0x0000.0000 (see page 586)																	
												UV3	UV2	UV1	UV0		
ADCSSPRI , type R/W, offset 0x020, reset 0x0000.3210 (see page 587)																	
												SS3	SS2	SS1	SS0		
ADCSPC , type R/W, offset 0x024, reset 0x0000.0000 (see page 589)																	
												SYNCWAIT			PHASE		
ADCPSSI , type R/W, offset 0x028, reset - (see page 590)																	
GSYNC												SS3	SS2	SS1	SS0		
ADCSAC , type R/W, offset 0x030, reset 0x0000.0000 (see page 592)																	
												Avg					
ADCDCISC , type R/W1C, offset 0x034, reset 0x0000.0000 (see page 593)																	
												DCINT7	DCINT6	DCINT5	DCINT4		
												DCINT3	DCINT2	DCINT1	DCINT0		
ADCCTL , type R/W, offset 0x038, reset 0x0000.0000 (see page 595)																	
												VREF					
ADCSSMUX0 , type R/W, offset 0x040, reset 0x0000.0000 (see page 596)																	
MUX7					MUX6				MUX5				MUX4				
MUX3					MUX2				MUX1				MUX0				
ADCSSCTL0 , type R/W, offset 0x044, reset 0x0000.0000 (see page 598)																	
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		
ADCSSFIFO0 , type RO, offset 0x048, reset - (see page 601)																	
												DATA					
ADCSSFIFO1 , type RO, offset 0x068, reset - (see page 601)																	
												DATA					
ADCSSFIFO2 , type RO, offset 0x088, reset - (see page 601)																	
												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												
ADCSS FIFO3, type RO, offset 0x0A8, reset - (see page 601)																											
DATA																											
ADCSS FIFO0, type RO, offset 0x04C, reset 0x0000.0100 (see page 602)																											
FULL				EMPTY				H PTR				T PTR															
ADCSS FIFO1, type RO, offset 0x06C, reset 0x0000.0100 (see page 602)																											
FULL				EMPTY				H PTR				T PTR															
ADCSS FIFO2, type RO, offset 0x08C, reset 0x0000.0100 (see page 602)																											
FULL				EMPTY				H PTR				T PTR															
ADCSS FIFO3, type RO, offset 0x0AC, reset 0x0000.0100 (see page 602)																											
S7DCOP				S6DCOP				S5DCOP				S4DCOP															
S3DCOP				S2DCOP				S1DCOP				S0DCOP															
ADCSS DCO 0, type R/W, offset 0x054, reset 0x0000.0000 (see page 604)																											
S7DCSEL				S6DCSEL				S5DCSEL				S4DCSEL															
S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL															
ADCSS MUX 1, type R/W, offset 0x060, reset 0x0000.0000 (see page 608)																											
MUX3				MUX2				MUX1				MUX0															
ADCSS MUX 2, type R/W, offset 0x080, reset 0x0000.0000 (see page 608)																											
MUX3				MUX2				MUX1				MUX0															
ADCSS CTL 1, type R/W, offset 0x064, reset 0x0000.0000 (see page 609)																											
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0												
ADCSS CTL 2, type R/W, offset 0x084, reset 0x0000.0000 (see page 609)																											
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0												
ADCSS OP 1, type R/W, offset 0x070, reset 0x0000.0000 (see page 611)																											
S3DCOP				S2DCOP				S1DCOP				S0DCOP															
ADCSS OP 2, type R/W, offset 0x090, reset 0x0000.0000 (see page 611)																											
S3DCOP				S2DCOP				S1DCOP				S0DCOP															
ADCSS DC 1, type R/W, offset 0x074, reset 0x0000.0000 (see page 612)																											
S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL															
ADCSS DC 2, type R/W, offset 0x094, reset 0x0000.0000 (see page 612)																											
S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL															
ADCSS MUX 3, type R/W, offset 0xA0, reset 0x0000.0000 (see page 614)																											
												MUX0															
ADCSS CTL 3, type R/W, offset 0xA4, reset 0x0000.0002 (see page 615)																											
												TS0	IE0	END0	D0												

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCSSOP3, type R/W, offset 0xB0, reset 0x0000.0000 (see page 616)															
S0DCOP															
ADCSSDC3, type R/W, offset 0xB4, reset 0x0000.0000 (see page 617)															
S0DCSEL															
ADCDCRIC, type R/W, offset 0xD00, reset 0x0000.0000 (see page 618)															
DCTRIG7 DCTRIG6 DCTRIG5 DCTRIG4 DCTRIG3 DCTRIG2 DCTRIG1 DCTRIG0 DCINT7 DCINT6 DCINT5 DCINT4 DCINT3 DCINT2 DCINT1 DCINT0															
ADCDCCCTL0, type R/W, offset 0xE00, reset 0x0000.0000 (see page 623)															
CTE CTC CTM CIE CIC CIM															
ADCDCCCTL1, type R/W, offset 0xE04, reset 0x0000.0000 (see page 623)															
CTE CTC CTM CIE CIC CIM															
ADCDCCCTL2, type R/W, offset 0xE08, reset 0x0000.0000 (see page 623)															
CTE CTC CTM CIE CIC CIM															
ADCDCCCTL3, type R/W, offset 0xE0C, reset 0x0000.0000 (see page 623)															
CTE CTC CTM CIE CIC CIM															
ADCDCCCTL4, type R/W, offset 0xE10, reset 0x0000.0000 (see page 623)															
CTE CTC CTM CIE CIC CIM															
ADCDCCCTL5, type R/W, offset 0xE14, reset 0x0000.0000 (see page 623)															
CTE CTC CTM CIE CIC CIM															
ADCDCCCTL6, type R/W, offset 0xE18, reset 0x0000.0000 (see page 623)															
CTE CTC CTM CIE CIC CIM															
ADCDCCCTL7, type R/W, offset 0xE1C, reset 0x0000.0000 (see page 623)															
CTE CTC CTM CIE CIC CIM															
ADCDCCMP0, type R/W, offset 0xE40, reset 0x0000.0000 (see page 626)															
COMP1 COMP0															
ADCDCCMP1, type R/W, offset 0xE44, reset 0x0000.0000 (see page 626)															
COMP1 COMP0															
ADCDCCMP2, type R/W, offset 0xE48, reset 0x0000.0000 (see page 626)															
COMP1 COMP0															
ADCDCCMP3, type R/W, offset 0xE4C, reset 0x0000.0000 (see page 626)															
COMP1 COMP0															
ADCDCCMP4, type R/W, offset 0xE50, reset 0x0000.0000 (see page 626)															
COMP1 COMP0															
ADCDCCMP5, type R/W, offset 0xE54, reset 0x0000.0000 (see page 626)															
COMP1 COMP0															

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
ADCDCMP6, type R/W, offset 0xE58, reset 0x0000.0000 (see page 626)															
														COMP1	
														COMP0	
ADCDCMP7, type R/W, offset 0xE5C, reset 0x0000.0000 (see page 626)															
														COMP1	
														COMP0	
Universal Asynchronous Receivers/Transmitters (UARTs)															
UART0 base: 0x4000.C000															
UART1 base: 0x4000.D000															
UART2 base: 0x4000.E000															
UARTDR, type R/W, offset 0x000, reset 0x0000.0000 (see page 641)															
				OE	BE	PE	FE							DATA	
UARTRSR/UARTECR, type RO, offset 0x004, reset 0x0000.0000 (Read-Only Status Register) (see page 643)															
														OE	BE
														PE	FE
UARTRSR/UARTECR, type WO, offset 0x004, reset 0x0000.0000 (Write-Only Error Clear Register) (see page 643)															
														DATA	
UARTFR, type RO, offset 0x018, reset 0x0000.0090 (see page 646)															
								RI	TXFE	RXFF	TXFF	RXFE	BUSY	DCD	DSR
															CTS
UARTILPR, type R/W, offset 0x020, reset 0x0000.0000 (see page 649)															
														ILPDVSR	
UARTIBRD, type R/W, offset 0x024, reset 0x0000.0000 (see page 650)															
														DIVINT	
UARTFBRD, type R/W, offset 0x028, reset 0x0000.0000 (see page 651)															
														DIVFRAC	
UARTLCRH, type R/W, offset 0x02C, reset 0x0000.0000 (see page 652)															
									SPS	WLEN	FEN	STP2	EPS	PEN	BRK
UARTCTL, type R/W, offset 0x030, reset 0x0000.0300 (see page 654)															
CTSEN	RTSEN			RTS	DTR	RXE	TXE	LBE	LIN	HSE	EOT	SMART	SIRLP	SIREN	UARTEN
UARTIFLS, type R/W, offset 0x034, reset 0x0000.0012 (see page 658)															
														RXIFLSEL	TXIFLSEL
UARTIM, type R/W, offset 0x038, reset 0x0000.0000 (see page 660)															
LME5IM	LME1IM	LMSBIM		OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM	DSRIM	DCDIM	CTSIM	RIIM	
UARTRIS, type RO, offset 0x03C, reset 0x0000.000F (see page 664)															
LME5RIS	LME1RIS	LMSBRIS		OERIS	BERIS	PERIS	FERIS	RTRIS	TXRIS	RXRIS	DSRRIS	DCDRIS	CTSRIS	RIRIS	
UARTMIS, type RO, offset 0x040, reset 0x0000.0000 (see page 667)															
LME5MIS	LME1MIS	LMSBMIS		OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS	DSRMIS	DCDMIS	CTSMIS	RIMIS	
UARTICR, type W1C, offset 0x044, reset 0x0000.0000 (see page 670)															
LME5MIC	LME1MIC	LMSBMC		OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC	DSRMIC	DCDMIC	CTSMIC	RIMIC	

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
UARTDMACTL , type R/W, offset 0x048, reset 0x0000.0000 (see page 672)															
													DMAERR	TXDMAE	RXDMAE
UARTLCTL , type R/W, offset 0x090, reset 0x0000.0000 (see page 673)															
													BLEN		MASTER
UARTLSS , type RO, offset 0x094, reset 0x0000.0000 (see page 674)															
													TSS		
UARTLTIM , type RO, offset 0x098, reset 0x0000.0000 (see page 675)															
															TIMER
UARTPeriphID4 , type RO, offset 0xFD0, reset 0x0000.0000 (see page 676)															
														PID4	
UARTPeriphID5 , type RO, offset 0xFD4, reset 0x0000.0000 (see page 677)															
													PID5		
UARTPeriphID6 , type RO, offset 0xFD8, reset 0x0000.0000 (see page 678)															
													PID6		
UARTPeriphID7 , type RO, offset 0xFDC, reset 0x0000.0000 (see page 679)															
													PID7		
UARTPeriphID0 , type RO, offset 0xFE0, reset 0x0000.0060 (see page 680)															
													PID0		
UARTPeriphID1 , type RO, offset 0xFE4, reset 0x0000.0000 (see page 681)															
													PID1		
UARTPeriphID2 , type RO, offset 0xFE8, reset 0x0000.0018 (see page 682)															
													PID2		
UARTPeriphID3 , type RO, offset 0xFEC, reset 0x0000.0001 (see page 683)															
													PID3		
UARTPCellID0 , type RO, offset 0xFF0, reset 0x0000.000D (see page 684)															
													CID0		
UARTPCellID1 , type RO, offset 0xFF4, reset 0x0000.00F0 (see page 685)															
													CID1		
UARTPCellID2 , type RO, offset 0xFF8, reset 0x0000.0005 (see page 686)															
													CID2		
UARTPCellID3 , type RO, offset 0xFFC, reset 0x0000.00B1 (see page 687)															
													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				
Synchronous Serial Interface (SSI)																			
SSI0 base: 0x4000.8000 SSI1 base: 0x4000.9000																			
SSICR0 , type R/W, offset 0x000, reset 0x0000.0000 (see page 704)																			
SCR								SPH	SPO	FRF	DSS								
SSICR1 , type R/W, offset 0x004, reset 0x0000.0000 (see page 706)																			
								EOT				SOD	MS	SSE	LBM				
SSIDR , type R/W, offset 0x008, reset 0x0000.0000 (see page 708)																			
DATA																			
SSISR , type RO, offset 0x00C, reset 0x0000.0003 (see page 709)																			
								BSY				RFF	RNE	TNF	TFE				
SSICPSR , type R/W, offset 0x010, reset 0x0000.0000 (see page 711)																			
								CPSDVSRR											
SSIIM , type R/W, offset 0x014, reset 0x0000.0000 (see page 712)																			
								TXIM				RXIM	RTIM	RORIM					
SSIRIS , type RO, offset 0x018, reset 0x0000.0008 (see page 713)																			
								TXRIS				RTRIS	RORRIS						
SSIMIS , type RO, offset 0x01C, reset 0x0000.0000 (see page 715)																			
								TXMIS				RXMIS	RTMIS	RORMIS					
SSIIICR , type W1C, offset 0x020, reset 0x0000.0000 (see page 717)																			
								RTIC											
SSIDMACTL , type R/W, offset 0x024, reset 0x0000.0000 (see page 718)																			
								TXDMAE				RXDMAE							
SSIPeriphID4 , type RO, offset 0xFD0, reset 0x0000.0000 (see page 719)																			
								PID4											
SSIPeriphID5 , type RO, offset 0xFD4, reset 0x0000.0000 (see page 720)																			
								PID5											
SSIPeriphID6 , type RO, offset 0xFD8, reset 0x0000.0000 (see page 721)																			
								PID6											
SSIPeriphID7 , type RO, offset 0xFDC, reset 0x0000.0000 (see page 722)																			
								PID7											
SSIPeriphID0 , type RO, offset 0xFE0, reset 0x0000.0022 (see page 723)																			
								PID0											
SSIPeriphID1 , type RO, offset 0xFE4, reset 0x0000.0000 (see page 724)																			
								PID1											

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSIPeriphID2 , type RO, offset 0xFE8, reset 0x0000.0018 (see page 725)															
PID2															
SSIPeriphID3 , type RO, offset 0xFEC, reset 0x0000.0001 (see page 726)															
PID3															
SSIPCellID0 , type RO, offset 0xFF0, reset 0x0000.000D (see page 727)															
CID0															
SSIPCellID1 , type RO, offset 0xFF4, reset 0x0000.00F0 (see page 728)															
CID1															
SSIPCellID2 , type RO, offset 0xFF8, reset 0x0000.0005 (see page 729)															
CID2															
SSIPCellID3 , type RO, offset 0xFFC, reset 0x0000.00B1 (see page 730)															
CID3															
Inter-Integrated Circuit (I²C) Interface															
I²C Master															
I ² C 0 base: 0x4002.0000															
I ² C 1 base: 0x4002.1000															
I²CMSA , type R/W, offset 0x000, reset 0x0000.0000															
SA															
I²CMCS , type RO, offset 0x004, reset 0x0000.0000 (Read-Only Status Register)															
BUSBSY IDLE ARBLST DATAACK ADRACK ERROR BUSY															
I²CMCS , type WO, offset 0x004, reset 0x0000.0000 (Write-Only Control Register)															
ACK STOP START RUN															
I²CMDR , type R/W, offset 0x008, reset 0x0000.0000															
DATA															
I²CMTPR , type R/W, offset 0x00C, reset 0x0000.0001															
TPR															
I²CMIMR , type R/W, offset 0x010, reset 0x0000.0000															
IM															
I²CMRIS , type RO, offset 0x014, reset 0x0000.0000															
RIS															
I²CMMS , type RO, offset 0x018, reset 0x0000.0000															
MIS															
I²CMICR , type WO, offset 0x01C, reset 0x0000.0000															
IC															
I²CMCR , type R/W, offset 0x020, reset 0x0000.0000															
SFE MFE LPBK															

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
Inter-Integrated Circuit (I²C) Interface															
I²C Slave															
I2C 0 base: 0x4002.0000 I2C 1 base: 0x4002.1000															
I2CSOAR, type R/W, offset 0x800, reset 0x0000.0000															
OAR															
I2CSCSR, type RO, offset 0x804, reset 0x0000.0000 (Read-Only Status Register)															
FBR TREQ RREQ															
I2CSCSR, type WO, offset 0x804, reset 0x0000.0000 (Write-Only Control Register)															
DA															
I2CSDR, type R/W, offset 0x808, reset 0x0000.0000															
DATA															
I2CSIMR, type R/W, offset 0x80C, reset 0x0000.0000															
STOPIM STARTIM DATAIM															
I2CSRIS, type RO, offset 0x810, reset 0x0000.0000															
STOPRIS STARTRIS DATARIS															
I2CSMIS, type RO, offset 0x814, reset 0x0000.0000															
STOPMIS STARTMIS DATAMIS															
I2CSICR, type WO, offset 0x818, reset 0x0000.0000															
STOPIC STARTIC DATAIC															
Inter-Integrated Circuit Sound (I²S) Interface															
Base 0x4005.4000															
I2STXFIFO, type WO, offset 0x000, reset 0x0000.0000 (see page 780)															
TXFIFO															
I2STXFIFOCFG, type R/W, offset 0x004, reset 0x0000.0000 (see page 781)															
CSS LRS															
I2STXCFG, type R/W, offset 0x008, reset 0x1400.7DF0 (see page 782)															
JST DLY SCP LRP WM FMT MSL SDSZ															
SSZ															
I2STXLIMIT, type R/W, offset 0x00C, reset 0x0000.0000 (see page 784)															
LIMIT															
I2STXISM, type R/W, offset 0x010, reset 0x0000.0000 (see page 785)															
FFI FFM															
I2STXLEV, type RO, offset 0x018, reset 0x0000.0000 (see page 786)															
LEVEL															
I2SRXFIFO, type RO, offset 0x800, reset 0x0000.0000 (see page 787)															
RXFIFO															
RXFIFO															

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
I2SRXIFOCFG , type R/W, offset 0x804, reset 0x0000.0000 (see page 788)															
I2SRXCFG , type R/W, offset 0x808, reset 0x1400.7DF0 (see page 789)															
I2SRXLIMIT , type R/W, offset 0x80C, reset 0x0000.7FFF (see page 791)															
I2SRXISM , type R/W, offset 0x810, reset 0x0000.0000 (see page 792)															
I2SRXLEV , type RO, offset 0x818, reset 0x0000.0000 (see page 793)															
I2SCFG , type R/W, offset 0xC00, reset 0x0000.0000 (see page 794)															
I2SIM , type R/W, offset 0xC10, reset 0x0000.0000 (see page 796)															
I2SRIS , type RO, offset 0xC14, reset 0x0000.0000 (see page 798)															
I2SMIS , type RO, offset 0xC18, reset 0x0000.0000 (see page 800)															
I2SIC , type WO, offset 0xC1C, reset 0x0000.0000 (see page 802)															
Controller Area Network (CAN) Module															
CAN0 base: 0x4004.0000															
CAN1 base: 0x4004.1000															
CANCTL , type R/W, offset 0x000, reset 0x0000.0001 (see page 824)															
CANSTS , type R/W, offset 0x004, reset 0x0000.0000 (see page 826)															
CANERR , type RO, offset 0x008, reset 0x0000.0000 (see page 829)															
CANBIT , type R/W, offset 0x00C, reset 0x0000.2301 (see page 830)															
CANINT , type RO, offset 0x010, reset 0x0000.0000 (see page 831)															
CANTST , type R/W, offset 0x014, reset 0x0000.0000 (see page 832)															

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
CANIF1DB1 , type R/W, offset 0x044, reset 0x0000.0000 (see page 848)															
DATA															
CANIF1DB2 , type R/W, offset 0x048, reset 0x0000.0000 (see page 848)															
DATA															
CANIF2DA1 , type R/W, offset 0x09C, reset 0x0000.0000 (see page 848)															
DATA															
CANIF2DA2 , type R/W, offset 0x0A0, reset 0x0000.0000 (see page 848)															
DATA															
CANIF2DB1 , type R/W, offset 0x0A4, reset 0x0000.0000 (see page 848)															
DATA															
CANIF2DB2 , type R/W, offset 0x0A8, reset 0x0000.0000 (see page 848)															
DATA															
CANTXRQ1 , type RO, offset 0x100, reset 0x0000.0000 (see page 849)															
TXRQST															
CANTXRQ2 , type RO, offset 0x104, reset 0x0000.0000 (see page 849)															
TXRQST															
CANNWDA1 , type RO, offset 0x120, reset 0x0000.0000 (see page 850)															
NEWDAT															
CANNWDA2 , type RO, offset 0x124, reset 0x0000.0000 (see page 850)															
NEWDAT															
CANMSG1INT , type RO, offset 0x140, reset 0x0000.0000 (see page 851)															
INTPND															
CANMSG2INT , type RO, offset 0x144, reset 0x0000.0000 (see page 851)															
INTPND															
CANMSG1VAL , type RO, offset 0x160, reset 0x0000.0000 (see page 852)															
MSGVAL															
CANMSG2VAL , type RO, offset 0x164, reset 0x0000.0000 (see page 852)															
MSGVAL															
Ethernet Controller															
Ethernet MAC (Ethernet Offset)															
Base 0x4004.8000															
MACRIS/MACIACK , type R/W1C, offset 0x000, reset 0x0000.0000															
PHYINT MDINT RXER FOV TXEMP TXER RXINT															
MACIM , type R/W, offset 0x004, reset 0x0000.007F															
PHYINTM MDINTM RXERM FOVM TXEMPM TXERM RXINTM															

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
MACRCTL, type R/W, offset 0x008, reset 0x0000.0008															
MACTCTL, type R/W, offset 0x00C, reset 0x0000.0000															
MACDATA, type RO, offset 0x010, reset 0x0000.0000 (Reads)															
MACDATA, type WO, offset 0x010, reset 0x0000.0000 (Writes)															
MACIA0, type R/W, offset 0x014, reset 0x0000.0000															
MACIA1, type R/W, offset 0x018, reset 0x0000.0000															
MACTHR, type R/W, offset 0x01C, reset 0x0000.003F															
MACMCTL, type R/W, offset 0x020, reset 0x0000.0000															
MACMDV, type R/W, offset 0x024, reset 0x0000.0080															
MACMTXD, type R/W, offset 0x02C, reset 0x0000.0000															
MACMRXD, type R/W, offset 0x030, reset 0x0000.0000															
MACNP, type RO, offset 0x034, reset 0x0000.0000															
MACTR, type R/W, offset 0x038, reset 0x0000.0000															
MACTS, type R/W, offset 0x03C, reset 0x0000.0000															
MACLED, type R/W, offset 0x040, reset 0x0000.0100															
MDIX, type R/W, offset 0x044, reset 0x0000.0000															
Ethernet Controller															
MII Management (Accessed through the MACMCTL register)															
MR0, type R/W, address 0x00, reset 0x1000															
RESET	LOOPBK	SPEEDSL	ANEGEN	PWRDN	ISO	RANEG	DUPLEX	COLT							

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												
MR1, type RO, address 0x01, reset 0x7809																											
100X_F	100X_H	10T_F	10T_H									ANEHC	RFAULT	ANEGA	LINK	JAB	EXTD										
MR2, type RO, address 0x02, reset 0x0161																											
OUI[21:6]																											
MR3, type RO, address 0x03, reset 0xB410																											
OUI[5:0]				MN				RN																			
MR4, type R/W, address 0x04, reset 0x01E1																											
NP		RF						A3	A2	A1	A0			S													
MR5, type RO, address 0x05, reset 0x0001																											
NP	ACK	RF		A				S																			
MR6, type RO, address 0x06, reset 0x0000																											
PDF								LPNPA				PRX	LPNEGA														
MR16, type RO, address 0x10, reset 0x0040																											
SR																											
MR17, type R/W, address 0x11, reset 0x0002																											
FASTRIP	EDPD		LSQE		FAASTEST								FGLS	ENON													
MR27, type RO, address 0x1B, reset -																											
XPOL																											
MR29, type RO, address 0x1D, reset 0x0000									EONIS	ANCOMPIS	RFLTIS	LDIS	LPACKIS	PDFIS	PRXIS												
EONIM								ANCOMPIM	RFLTIM	LDIM	LPACKIM	PDFIM	PRXIM														
MR31, type R/W, address 0x1F, reset 0x0040																											
AUTODONE								SPEED				SCRDIS															
Universal Serial Bus (USB) Controller																											
Base 0x4005.0000																											
USBFADDR, type R/W, offset 0x000, reset 0x00 (see page 942)																											
FUNCADDR																											
USBPOWER, type R/W, offset 0x001, reset 0x20 (OTG A / Host Mode) (see page 943)																											
RESET								RESUME	SUSPEND	PWRDNPHY																	
USBPOWER, type R/W, offset 0x001, reset 0x20 (OTG B / Device Mode) (see page 943)																											
ISOUP								RESET	RESUME	SUSPEND	PWRDNPHY																
USBTXIS, type RO, offset 0x002, reset 0x0000 (see page 946)																											
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0												
USBRXIS, type RO, offset 0x004, reset 0x0000 (see page 948)																											
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1													
USBTXIE, type R/W, offset 0x006, reset 0xFFFF (see page 950)																											
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1	EP0												
USBRXIE, type R/W, offset 0x008, reset 0xFFFF (see page 952)																											
EP15	EP14	EP13	EP12	EP11	EP10	EP9	EP8	EP7	EP6	EP5	EP4	EP3	EP2	EP1													
USBIS, type RO, offset 0x0A, reset 0x00 (OTG A / Host Mode) (see page 954)																											
VBUSERR								SESREQ	DISCON	CONN	SOF	BABBLE	RESUME														
USBIS, type RO, offset 0x0A, reset 0x00 (OTG B / Device Mode) (see page 954)								DISCON				SOF	RESET	RESUME	SUSPEND												
USBIE, type R/W, offset 0x0B, reset 0x06 (OTG A / Host Mode) (see page 957)																											
VBUSERR								SESREQ	DISCON	CONN	SOF	BABBLE	RESUME														
USBIE, type R/W, offset 0x0B, reset 0x06 (OTG B / Device Mode) (see page 957)								DISCON				SOF	RESET	RESUME	SUSPEND												
USBFRAME, type RO, offset 0x0C, reset 0x0000 (see page 960)																											
FRAME																											
USBEPIIDX, type R/W, offset 0x0E, reset 0x00 (see page 961)																											
EPIIDX																											

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
USBTEST, type R/W, offset 0x00F, reset 0x00 (OTG A / Host Mode) (see page 962)															
								FORCEH	FIFOACC	FORCEFS					
USBTEST, type R/W, offset 0x00F, reset 0x00 (OTG B / Device Mode) (see page 962)															
								FIFOACC	FORCEFS						
USBFIFO0, type R/W, offset 0x020, reset 0x0000.0000 (see page 964)															
								EPDATA							
								EPDATA							
USBFIFO1, type R/W, offset 0x024, reset 0x0000.0000 (see page 964)															
								EPDATA							
								EPDATA							
USBFIFO2, type R/W, offset 0x028, reset 0x0000.0000 (see page 964)															
								EPDATA							
								EPDATA							
USBFIFO3, type R/W, offset 0x02C, reset 0x0000.0000 (see page 964)															
								EPDATA							
								EPDATA							
USBFIFO4, type R/W, offset 0x030, reset 0x0000.0000 (see page 964)															
								EPDATA							
								EPDATA							
USBFIFO5, type R/W, offset 0x034, reset 0x0000.0000 (see page 964)															
								EPDATA							
								EPDATA							
USBFIFO6, type R/W, offset 0x038, reset 0x0000.0000 (see page 964)															
								EPDATA							
								EPDATA							
USBFIFO7, type R/W, offset 0x03C, reset 0x0000.0000 (see page 964)															
								EPDATA							
								EPDATA							
USBFIFO8, type R/W, offset 0x040, reset 0x0000.0000 (see page 964)															
								EPDATA							
								EPDATA							
USBFIFO9, type R/W, offset 0x044, reset 0x0000.0000 (see page 964)															
								EPDATA							
								EPDATA							
USBFIFO10, type R/W, offset 0x048, reset 0x0000.0000 (see page 964)															
								EPDATA							
								EPDATA							
USBFIFO11, type R/W, offset 0x04C, reset 0x0000.0000 (see page 964)															
								EPDATA							
								EPDATA							
USBFIFO12, type R/W, offset 0x050, reset 0x0000.0000 (see page 964)															
								EPDATA							
								EPDATA							
USBFIFO13, type R/W, offset 0x054, reset 0x0000.0000 (see page 964)															
								EPDATA							
								EPDATA							
USBFIFO14, type R/W, offset 0x058, reset 0x0000.0000 (see page 964)															
								EPDATA							
								EPDATA							
USBFIFO15, type R/W, offset 0x05C, reset 0x0000.0000 (see page 964)															
								EPDATA							
								EPDATA							

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
USBDEVCTL , type R/W, offset 0x060, reset 0x80 (see page 966)															
								DEV	FSDEV	LSDEV	VBUS	HOST	HOSTREQ	SESSION	
USBTXFIFOSZ , type R/W, offset 0x062, reset 0x00 (see page 968)															
										DPB					SIZE
USBRXFIFOSZ , type R/W, offset 0x063, reset 0x00 (see page 968)															
										DPB					SIZE
USBTXFIFOADD , type R/W, offset 0x064, reset 0x0000 (see page 969)															
															ADDR
USBRXFIFOADD , type R/W, offset 0x066, reset 0x0000 (see page 969)															
															ADDR
USBCONTIM , type R/W, offset 0x07A, reset 0x5C (see page 970)															
											WTCON				WTID
USBVPLEN , type R/W, offset 0x07B, reset 0x3C (see page 971)															
															VPLEN
USBFSEOF , type R/W, offset 0x07D, reset 0x77 (see page 972)															
															FSEOFG
USBLSEOF , type R/W, offset 0x07E, reset 0x72 (see page 973)															
															LSEOFG
USBTXFUNCADDR0 , type R/W, offset 0x080, reset 0x00 (see page 974)															
															ADDR
USBTXFUNCADDR1 , type R/W, offset 0x088, reset 0x00 (see page 974)															
															ADDR
USBTXFUNCADDR2 , type R/W, offset 0x090, reset 0x00 (see page 974)															
															ADDR
USBTXFUNCADDR3 , type R/W, offset 0x098, reset 0x00 (see page 974)															
															ADDR
USBTXFUNCADDR4 , type R/W, offset 0x0A0, reset 0x00 (see page 974)															
															ADDR
USBTXFUNCADDR5 , type R/W, offset 0x0A8, reset 0x00 (see page 974)															
															ADDR
USBTXFUNCADDR6 , type R/W, offset 0x0B0, reset 0x00 (see page 974)															
															ADDR
USBTXFUNCADDR7 , type R/W, offset 0x0B8, reset 0x00 (see page 974)															
															ADDR
USBTXFUNCADDR8 , type R/W, offset 0x0C0, reset 0x00 (see page 974)															
															ADDR
USBTXFUNCADDR9 , type R/W, offset 0x0C8, reset 0x00 (see page 974)															
															ADDR
USBTXFUNCADDR10 , type R/W, offset 0x0D0, reset 0x00 (see page 974)															
															ADDR
USBTXFUNCADDR11 , type R/W, offset 0x0D8, reset 0x00 (see page 974)															
															ADDR
USBTXFUNCADDR12 , type R/W, offset 0x0E0, reset 0x00 (see page 974)															
															ADDR
USBTXFUNCADDR13 , type R/W, offset 0x0E8, reset 0x00 (see page 974)															
															ADDR
USBTXFUNCADDR14 , type R/W, offset 0x0F0, reset 0x00 (see page 974)															
															ADDR
USBTXFUNCADDR15 , type R/W, offset 0x0F8, reset 0x00 (see page 974)															
															ADDR
USBTXHUBADDR0 , type R/W, offset 0x082, reset 0x00 (see page 976)															
								MULTTRAN							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
USBTXHUBADDR1 , type R/W, offset 0x08A, reset 0x00 (see page 976)															
												MULTTRAN	ADDR		
USBTXHUBADDR2 , type R/W, offset 0x092, reset 0x00 (see page 976)															
												MULTTRAN	ADDR		
USBTXHUBADDR3 , type R/W, offset 0x09A, reset 0x00 (see page 976)															
												MULTTRAN	ADDR		
USBTXHUBADDR4 , type R/W, offset 0x0A2, reset 0x00 (see page 976)															
												MULTTRAN	ADDR		
USBTXHUBADDR5 , type R/W, offset 0x0AA, reset 0x00 (see page 976)															
												MULTTRAN	ADDR		
USBTXHUBADDR6 , type R/W, offset 0x0B2, reset 0x00 (see page 976)															
												MULTTRAN	ADDR		
USBTXHUBADDR7 , type R/W, offset 0x0BA, reset 0x00 (see page 976)															
												MULTTRAN	ADDR		
USBTXHUBADDR8 , type R/W, offset 0x0C2, reset 0x00 (see page 976)															
												MULTTRAN	ADDR		
USBTXHUBADDR9 , type R/W, offset 0x0CA, reset 0x00 (see page 976)															
												MULTTRAN	ADDR		
USBTXHUBADDR10 , type R/W, offset 0x0D2, reset 0x00 (see page 976)															
												MULTTRAN	ADDR		
USBTXHUBADDR11 , type R/W, offset 0x0DA, reset 0x00 (see page 976)															
												MULTTRAN	ADDR		
USBTXHUBADDR12 , type R/W, offset 0x0E2, reset 0x00 (see page 976)															
												MULTTRAN	ADDR		
USBTXHUBADDR13 , type R/W, offset 0x0EA, reset 0x00 (see page 976)															
												MULTTRAN	ADDR		
USBTXHUBADDR14 , type R/W, offset 0x0F2, reset 0x00 (see page 976)															
												MULTTRAN	ADDR		
USBTXHUBADDR15 , type R/W, offset 0x0FA, reset 0x00 (see page 976)															
												MULTTRAN	ADDR		
USBTXHUBPORT0 , type R/W, offset 0x083, reset 0x00 (see page 978)															
													PORT		
USBTXHUBPORT1 , type R/W, offset 0x08B, reset 0x00 (see page 978)															
													PORT		
USBTXHUBPORT2 , type R/W, offset 0x093, reset 0x00 (see page 978)															
													PORT		
USBTXHUBPORT3 , type R/W, offset 0x09B, reset 0x00 (see page 978)															
													PORT		
USBTXHUBPORT4 , type R/W, offset 0x0A3, reset 0x00 (see page 978)															
													PORT		
USBTXHUBPORT5 , type R/W, offset 0x0AB, reset 0x00 (see page 978)															
													PORT		
USBTXHUBPORT6 , type R/W, offset 0x0B3, reset 0x00 (see page 978)															
													PORT		
USBTXHUBPORT7 , type R/W, offset 0x0BB, reset 0x00 (see page 978)															
													PORT		
USBTXHUBPORT8 , type R/W, offset 0x0C3, reset 0x00 (see page 978)															
													PORT		
USBTXHUBPORT9 , type R/W, offset 0x0CB, reset 0x00 (see page 978)															
													PORT		
USBTXHUBPORT10 , type R/W, offset 0x0D3, reset 0x00 (see page 978)															
													PORT		

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
USBTXHUBPORT11, type R/W, offset 0x0DB, reset 0x00 (see page 978)															
															PORT
USBTXHUBPORT12, type R/W, offset 0x0E3, reset 0x00 (see page 978)															
															PORT
USBTXHUBPORT13, type R/W, offset 0x0EB, reset 0x00 (see page 978)															
															PORT
USBTXHUBPORT14, type R/W, offset 0x0F3, reset 0x00 (see page 978)															
															PORT
USBTXHUBPORT15, type R/W, offset 0x0FB, reset 0x00 (see page 978)															
															PORT
USBRXFUNCADDR1, type R/W, offset 0x08C, reset 0x00 (see page 980)															
															ADDR
USBRXFUNCADDR2, type R/W, offset 0x094, reset 0x00 (see page 980)															
															ADDR
USBRXFUNCADDR3, type R/W, offset 0x09C, reset 0x00 (see page 980)															
															ADDR
USBRXFUNCADDR4, type R/W, offset 0xA4, reset 0x00 (see page 980)															
															ADDR
USBRXFUNCADDR5, type R/W, offset 0x0AC, reset 0x00 (see page 980)															
															ADDR
USBRXFUNCADDR6, type R/W, offset 0x0B4, reset 0x00 (see page 980)															
															ADDR
USBRXFUNCADDR7, type R/W, offset 0x0BC, reset 0x00 (see page 980)															
															ADDR
USBRXFUNCADDR8, type R/W, offset 0x0C4, reset 0x00 (see page 980)															
															ADDR
USBRXFUNCADDR9, type R/W, offset 0x0CC, reset 0x00 (see page 980)															
															ADDR
USBRXFUNCADDR10, type R/W, offset 0xD4, reset 0x00 (see page 980)															
															ADDR
USBRXFUNCADDR11, type R/W, offset 0xDC, reset 0x00 (see page 980)															
															ADDR
USBRXFUNCADDR12, type R/W, offset 0xE4, reset 0x00 (see page 980)															
															ADDR
USBRXFUNCADDR13, type R/W, offset 0xEC, reset 0x00 (see page 980)															
															ADDR
USBRXFUNCADDR14, type R/W, offset 0xF4, reset 0x00 (see page 980)															
															ADDR
USBRXFUNCADDR15, type R/W, offset 0xFC, reset 0x00 (see page 980)															
															ADDR
USBRXHUBADDR1, type R/W, offset 0x08E, reset 0x00 (see page 982)															
													MULTTRAN		ADDR
USBRXHUBADDR2, type R/W, offset 0x096, reset 0x00 (see page 982)															
													MULTTRAN		ADDR
USBRXHUBADDR3, type R/W, offset 0x09E, reset 0x00 (see page 982)															
													MULTTRAN		ADDR
USBRXHUBADDR4, type R/W, offset 0xA6, reset 0x00 (see page 982)															
													MULTTRAN		ADDR
USBRXHUBADDR5, type R/W, offset 0xAE, reset 0x00 (see page 982)															
													MULTTRAN		ADDR
USBRXHUBADDR6, type R/W, offset 0xB6, reset 0x00 (see page 982)															
													MULTTRAN		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
USBRXHUBADDR7 , type R/W, offset 0x0BE, reset 0x00 (see page 982)															
												MULTTRAN	ADDR		
USBRXHUBADDR8 , type R/W, offset 0x0C6, reset 0x00 (see page 982)															
												MULTTRAN	ADDR		
USBRXHUBADDR9 , type R/W, offset 0x0CE, reset 0x00 (see page 982)															
												MULTTRAN	ADDR		
USBRXHUBADDR10 , type R/W, offset 0x0D6, reset 0x00 (see page 982)															
												MULTTRAN	ADDR		
USBRXHUBADDR11 , type R/W, offset 0x0DE, reset 0x00 (see page 982)															
												MULTTRAN	ADDR		
USBRXHUBADDR12 , type R/W, offset 0x0E6, reset 0x00 (see page 982)															
												MULTTRAN	ADDR		
USBRXHUBADDR13 , type R/W, offset 0x0EE, reset 0x00 (see page 982)															
												MULTTRAN	ADDR		
USBRXHUBADDR14 , type R/W, offset 0x0F6, reset 0x00 (see page 982)															
												MULTTRAN	ADDR		
USBRXHUBADDR15 , type R/W, offset 0x0FE, reset 0x00 (see page 982)															
												MULTTRAN	ADDR		
USBRXHUBPORT1 , type R/W, offset 0x08F, reset 0x00 (see page 984)															
													PORT		
USBRXHUBPORT2 , type R/W, offset 0x097, reset 0x00 (see page 984)															
													PORT		
USBRXHUBPORT3 , type R/W, offset 0x09F, reset 0x00 (see page 984)															
													PORT		
USBRXHUBPORT4 , type R/W, offset 0x0A7, reset 0x00 (see page 984)															
													PORT		
USBRXHUBPORT5 , type R/W, offset 0x0AF, reset 0x00 (see page 984)															
													PORT		
USBRXHUBPORT6 , type R/W, offset 0x0B7, reset 0x00 (see page 984)															
													PORT		
USBRXHUBPORT7 , type R/W, offset 0x0BF, reset 0x00 (see page 984)															
													PORT		
USBRXHUBPORT8 , type R/W, offset 0x0C7, reset 0x00 (see page 984)															
													PORT		
USBRXHUBPORT9 , type R/W, offset 0x0CF, reset 0x00 (see page 984)															
													PORT		
USBRXHUBPORT10 , type R/W, offset 0x0D7, reset 0x00 (see page 984)															
													PORT		
USBRXHUBPORT11 , type R/W, offset 0x0DF, reset 0x00 (see page 984)															
													PORT		
USBRXHUBPORT12 , type R/W, offset 0x0E7, reset 0x00 (see page 984)															
													PORT		
USBRXHUBPORT13 , type R/W, offset 0x0EF, reset 0x00 (see page 984)															
													PORT		
USBRXHUBPORT14 , type R/W, offset 0x0F7, reset 0x00 (see page 984)															
													PORT		
USBRXHUBPORT15 , type R/W, offset 0x0FF, reset 0x00 (see page 984)															
													PORT		
USBTXMAXP1 , type R/W, offset 0x110, reset 0x0000 (see page 986)															
													MAXLOAD		
USBTXMAXP2 , type R/W, offset 0x120, reset 0x0000 (see page 986)															
													MAXLOAD		

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
USBTXMAXP3, type R/W, offset 0x130, reset 0x0000 (see page 986)															
															MAXLOAD
USBTXMAXP4, type R/W, offset 0x140, reset 0x0000 (see page 986)															
															MAXLOAD
USBTXMAXP5, type R/W, offset 0x150, reset 0x0000 (see page 986)															
															MAXLOAD
USBTXMAXP6, type R/W, offset 0x160, reset 0x0000 (see page 986)															
															MAXLOAD
USBTXMAXP7, type R/W, offset 0x170, reset 0x0000 (see page 986)															
															MAXLOAD
USBTXMAXP8, type R/W, offset 0x180, reset 0x0000 (see page 986)															
															MAXLOAD
USBTXMAXP9, type R/W, offset 0x190, reset 0x0000 (see page 986)															
															MAXLOAD
USBTXMAXP10, type R/W, offset 0x1A0, reset 0x0000 (see page 986)															
															MAXLOAD
USBTXMAXP11, type R/W, offset 0x1B0, reset 0x0000 (see page 986)															
															MAXLOAD
USBTXMAXP12, type R/W, offset 0x1C0, reset 0x0000 (see page 986)															
															MAXLOAD
USBTXMAXP13, type R/W, offset 0x1D0, reset 0x0000 (see page 986)															
															MAXLOAD
USBTXMAXP14, type R/W, offset 0x1E0, reset 0x0000 (see page 986)															
															MAXLOAD
USBTXMAXP15, type R/W, offset 0x1F0, reset 0x0000 (see page 986)															
															MAXLOAD
USBCSRL0, type W1C, offset 0x102, reset 0x00 (OTG A / Host Mode) (see page 988)															
															NAKTO STATUS REQPKT ERROR SETUP STALLED TXRDY RXRDY
USBCSRL0, type W1C, offset 0x102, reset 0x00 (OTG B / Device Mode) (see page 988)															
															SETENDC RXRDYC STALL SETEND DATAEND STALLED TXRDY RXRDY
USBCSRH0, type W1C, offset 0x103, reset 0x00 (OTG A / Host Mode) (see page 992)															
															DTWE DT FLUSH
USBCSRH0, type W1C, offset 0x103, reset 0x00 (OTG B / Device Mode) (see page 992)															
															FLUSH
USBCOUNT0, type RO, offset 0x108, reset 0x00 (see page 994)															
															COUNT
USBTYPE0, type R/W, offset 0x10A, reset 0x00 (see page 995)															
															SPEED
USBNAKLMT, type R/W, offset 0x10B, reset 0x00 (see page 996)															
															NAKLMT
USBTXCSRL1, type R/W, offset 0x112, reset 0x00 (OTG A / Host Mode) (see page 997)															
															NAKTO CLRDT STALLED SETUP FLUSH ERROR FIFONE TXRDY
USBTXCSRL2, type R/W, offset 0x122, reset 0x00 (OTG A / Host Mode) (see page 997)															
															NAKTO CLRDT STALLED SETUP FLUSH ERROR FIFONE TXRDY
USBTXCSRL3, type R/W, offset 0x132, reset 0x00 (OTG A / Host Mode) (see page 997)															
															NAKTO CLRDT STALLED SETUP FLUSH ERROR FIFONE TXRDY
USBTXCSRL4, type R/W, offset 0x142, reset 0x00 (OTG A / Host Mode) (see page 997)															
															NAKTO CLRDT STALLED SETUP FLUSH ERROR FIFONE TXRDY
USBTXCSRL5, type R/W, offset 0x152, reset 0x00 (OTG A / Host Mode) (see page 997)															
															NAKTO CLRDT STALLED SETUP FLUSH ERROR FIFONE TXRDY
USBTXCSRL6, type R/W, offset 0x162, reset 0x00 (OTG A / Host Mode) (see page 997)															
															NAKTO CLRDT STALLED SETUP FLUSH ERROR FIFONE TXRDY

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	
USBTXCSR7, type R/W, offset 0x172, reset 0x00 (OTG A / Host Mode) (see page 997)																
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY	
USBTXCSR8, type R/W, offset 0x182, reset 0x00 (OTG A / Host Mode) (see page 997)																
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY	
USBTXCSR9, type R/W, offset 0x192, reset 0x00 (OTG A / Host Mode) (see page 997)																
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY	
USBTXCSR10, type R/W, offset 0x1A2, reset 0x00 (OTG A / Host Mode) (see page 997)																
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY	
USBTXCSR11, type R/W, offset 0x1B2, reset 0x00 (OTG A / Host Mode) (see page 997)																
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY	
USBTXCSR12, type R/W, offset 0x1C2, reset 0x00 (OTG A / Host Mode) (see page 997)																
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY	
USBTXCSR13, type R/W, offset 0x1D2, reset 0x00 (OTG A / Host Mode) (see page 997)																
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY	
USBTXCSR14, type R/W, offset 0x1E2, reset 0x00 (OTG A / Host Mode) (see page 997)																
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY	
USBTXCSR15, type R/W, offset 0x1F2, reset 0x00 (OTG A / Host Mode) (see page 997)																
								NAKTO	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY	
USBTXCSR1, type R/W, offset 0x112, reset 0x00 (OTG B / Device Mode) (see page 997)																
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY	
USBTXCSR2, type R/W, offset 0x122, reset 0x00 (OTG B / Device Mode) (see page 997)																
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY	
USBTXCSR3, type R/W, offset 0x132, reset 0x00 (OTG B / Device Mode) (see page 997)																
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY	
USBTXCSR4, type R/W, offset 0x142, reset 0x00 (OTG B / Device Mode) (see page 997)																
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY	
USBTXCSR5, type R/W, offset 0x152, reset 0x00 (OTG B / Device Mode) (see page 997)																
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY	
USBTXCSR6, type R/W, offset 0x162, reset 0x00 (OTG B / Device Mode) (see page 997)																
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY	
USBTXCSR7, type R/W, offset 0x172, reset 0x00 (OTG B / Device Mode) (see page 997)																
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY	
USBTXCSR8, type R/W, offset 0x182, reset 0x00 (OTG B / Device Mode) (see page 997)																
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY	
USBTXCSR9, type R/W, offset 0x192, reset 0x00 (OTG B / Device Mode) (see page 997)																
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY	
USBTXCSR10, type R/W, offset 0x1A2, reset 0x00 (OTG B / Device Mode) (see page 997)																
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY	
USBTXCSR11, type R/W, offset 0x1B2, reset 0x00 (OTG B / Device Mode) (see page 997)																
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY	
USBTXCSR12, type R/W, offset 0x1C2, reset 0x00 (OTG B / Device Mode) (see page 997)																
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY	
USBTXCSR13, type R/W, offset 0x1D2, reset 0x00 (OTG B / Device Mode) (see page 997)																
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY	
USBTXCSR14, type R/W, offset 0x1E2, reset 0x00 (OTG B / Device Mode) (see page 997)																
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY	
USBTXCSR15, type R/W, offset 0x1F2, reset 0x00 (OTG B / Device Mode) (see page 997)																
									CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY	
USBTCSR1, type R/W, offset 0x113, reset 0x00 (OTG A / Host Mode) (see page 1002)																
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT	
USBTCSR2, type R/W, offset 0x123, reset 0x00 (OTG A / Host Mode) (see page 1002)									AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT

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
USBTXCSRH3, type R/W, offset 0x133, reset 0x00 (OTG A / Host Mode) (see page 1002)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH4, type R/W, offset 0x143, reset 0x00 (OTG A / Host Mode) (see page 1002)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH5, type R/W, offset 0x153, reset 0x00 (OTG A / Host Mode) (see page 1002)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH6, type R/W, offset 0x163, reset 0x00 (OTG A / Host Mode) (see page 1002)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH7, type R/W, offset 0x173, reset 0x00 (OTG A / Host Mode) (see page 1002)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH8, type R/W, offset 0x183, reset 0x00 (OTG A / Host Mode) (see page 1002)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH9, type R/W, offset 0x193, reset 0x00 (OTG A / Host Mode) (see page 1002)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH10, type R/W, offset 0x1A3, reset 0x00 (OTG A / Host Mode) (see page 1002)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH11, type R/W, offset 0x1B3, reset 0x00 (OTG A / Host Mode) (see page 1002)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH12, type R/W, offset 0x1C3, reset 0x00 (OTG A / Host Mode) (see page 1002)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH13, type R/W, offset 0x1D3, reset 0x00 (OTG A / Host Mode) (see page 1002)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH14, type R/W, offset 0x1E3, reset 0x00 (OTG A / Host Mode) (see page 1002)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH15, type R/W, offset 0x1F3, reset 0x00 (OTG A / Host Mode) (see page 1002)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH1, type R/W, offset 0x113, reset 0x00 (OTG B / Device Mode) (see page 1002)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH2, type R/W, offset 0x123, reset 0x00 (OTG B / Device Mode) (see page 1002)									AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH3, type R/W, offset 0x133, reset 0x00 (OTG B / Device Mode) (see page 1002)									AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH4, type R/W, offset 0x143, reset 0x00 (OTG B / Device Mode) (see page 1002)									AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH5, type R/W, offset 0x153, reset 0x00 (OTG B / Device Mode) (see page 1002)									AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH6, type R/W, offset 0x163, reset 0x00 (OTG B / Device Mode) (see page 1002)									AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH7, type R/W, offset 0x173, reset 0x00 (OTG B / Device Mode) (see page 1002)									AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH8, type R/W, offset 0x183, reset 0x00 (OTG B / Device Mode) (see page 1002)									AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH9, type R/W, offset 0x193, reset 0x00 (OTG B / Device Mode) (see page 1002)									AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH10, type R/W, offset 0x1A3, reset 0x00 (OTG B / Device Mode) (see page 1002)									AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH11, type R/W, offset 0x1B3, reset 0x00 (OTG B / Device Mode) (see page 1002)									AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH12, type R/W, offset 0x1C3, reset 0x00 (OTG B / Device Mode) (see page 1002)									AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH13, type R/W, offset 0x1D3, reset 0x00 (OTG B / Device Mode) (see page 1002)									AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		

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
USBTXCSRH14 , type R/W, offset 0x1E3, reset 0x00 (OTG B / Device Mode) (see page 1002)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH15 , type R/W, offset 0x1F3, reset 0x00 (OTG B / Device Mode) (see page 1002)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBRXMAXP1 , type R/W, offset 0x114, reset 0x0000 (see page 1006)															
									MAXLOAD						
USBRXMAXP2 , type R/W, offset 0x124, reset 0x0000 (see page 1006)															
									MAXLOAD						
USBRXMAXP3 , type R/W, offset 0x134, reset 0x0000 (see page 1006)															
									MAXLOAD						
USBRXMAXP4 , type R/W, offset 0x144, reset 0x0000 (see page 1006)															
									MAXLOAD						
USBRXMAXP5 , type R/W, offset 0x154, reset 0x0000 (see page 1006)															
									MAXLOAD						
USBRXMAXP6 , type R/W, offset 0x164, reset 0x0000 (see page 1006)															
									MAXLOAD						
USBRXMAXP7 , type R/W, offset 0x174, reset 0x0000 (see page 1006)															
									MAXLOAD						
USBRXMAXP8 , type R/W, offset 0x184, reset 0x0000 (see page 1006)															
									MAXLOAD						
USBRXMAXP9 , type R/W, offset 0x194, reset 0x0000 (see page 1006)															
									MAXLOAD						
USBRXMAXP10 , type R/W, offset 0x1A4, reset 0x0000 (see page 1006)															
									MAXLOAD						
USBRXMAXP11 , type R/W, offset 0x1B4, reset 0x0000 (see page 1006)															
									MAXLOAD						
USBRXMAXP12 , type R/W, offset 0x1C4, reset 0x0000 (see page 1006)															
									MAXLOAD						
USBRXMAXP13 , type R/W, offset 0x1D4, reset 0x0000 (see page 1006)															
									MAXLOAD						
USBRXMAXP14 , type R/W, offset 0x1E4, reset 0x0000 (see page 1006)															
									MAXLOAD						
USBRXMAXP15 , type R/W, offset 0x1F4, reset 0x0000 (see page 1006)															
									MAXLOAD						
USBRXCRL1 , type R/W, offset 0x116, reset 0x00 (OTG A / Host Mode) (see page 1008)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCRL2 , type R/W, offset 0x126, reset 0x00 (OTG A / Host Mode) (see page 1008)								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCRL3 , type R/W, offset 0x136, reset 0x00 (OTG A / Host Mode) (see page 1008)								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCRL4 , type R/W, offset 0x146, reset 0x00 (OTG A / Host Mode) (see page 1008)								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCRL5 , type R/W, offset 0x156, reset 0x00 (OTG A / Host Mode) (see page 1008)								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCRL6 , type R/W, offset 0x166, reset 0x00 (OTG A / Host Mode) (see page 1008)								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCRL7 , type R/W, offset 0x176, reset 0x00 (OTG A / Host Mode) (see page 1008)								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY

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
USBRXCSRL8, type R/W, offset 0x186, reset 0x00 (OTG A / Host Mode) (see page 1008)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL9, type R/W, offset 0x196, reset 0x00 (OTG A / Host Mode) (see page 1008)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL10, type R/W, offset 0x1A6, reset 0x00 (OTG A / Host Mode) (see page 1008)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL11, type R/W, offset 0x1B6, reset 0x00 (OTG A / Host Mode) (see page 1008)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL12, type R/W, offset 0x1C6, reset 0x00 (OTG A / Host Mode) (see page 1008)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL13, type R/W, offset 0x1D6, reset 0x00 (OTG A / Host Mode) (see page 1008)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL14, type R/W, offset 0x1E6, reset 0x00 (OTG A / Host Mode) (see page 1008)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL15, type R/W, offset 0x1F6, reset 0x00 (OTG A / Host Mode) (see page 1008)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL1, type R/W, offset 0x116, reset 0x00 (OTG B / Device Mode) (see page 1008)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL2, type R/W, offset 0x126, reset 0x00 (OTG B / Device Mode) (see page 1008)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL3, type R/W, offset 0x136, reset 0x00 (OTG B / Device Mode) (see page 1008)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL4, type R/W, offset 0x146, reset 0x00 (OTG B / Device Mode) (see page 1008)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL5, type R/W, offset 0x156, reset 0x00 (OTG B / Device Mode) (see page 1008)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL6, type R/W, offset 0x166, reset 0x00 (OTG B / Device Mode) (see page 1008)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL7, type R/W, offset 0x176, reset 0x00 (OTG B / Device Mode) (see page 1008)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL8, type R/W, offset 0x186, reset 0x00 (OTG B / Device Mode) (see page 1008)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL9, type R/W, offset 0x196, reset 0x00 (OTG B / Device Mode) (see page 1008)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL10, type R/W, offset 0x1A6, reset 0x00 (OTG B / Device Mode) (see page 1008)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL11, type R/W, offset 0x1B6, reset 0x00 (OTG B / Device Mode) (see page 1008)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL12, type R/W, offset 0x1C6, reset 0x00 (OTG B / Device Mode) (see page 1008)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL13, type R/W, offset 0x1D6, reset 0x00 (OTG B / Device Mode) (see page 1008)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL14, type R/W, offset 0x1E6, reset 0x00 (OTG B / Device Mode) (see page 1008)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL15, type R/W, offset 0x1F6, reset 0x00 (OTG B / Device Mode) (see page 1008)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY

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
USBRXCSRH1, type R/W, offset 0x117, reset 0x00 (OTG A / Host Mode) (see page 1013)															
					AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT				
USBRXCSRH2, type R/W, offset 0x127, reset 0x00 (OTG A / Host Mode) (see page 1013)															
					AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT				
USBRXCSRH3, type R/W, offset 0x137, reset 0x00 (OTG A / Host Mode) (see page 1013)															
					AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT				
USBRXCSRH4, type R/W, offset 0x147, reset 0x00 (OTG A / Host Mode) (see page 1013)															
					AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT				
USBRXCSRH5, type R/W, offset 0x157, reset 0x00 (OTG A / Host Mode) (see page 1013)															
					AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT				
USBRXCSRH6, type R/W, offset 0x167, reset 0x00 (OTG A / Host Mode) (see page 1013)															
					AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT				
USBRXCSRH7, type R/W, offset 0x177, reset 0x00 (OTG A / Host Mode) (see page 1013)															
					AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT				
USBRXCSRH8, type R/W, offset 0x187, reset 0x00 (OTG A / Host Mode) (see page 1013)															
					AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT				
USBRXCSRH9, type R/W, offset 0x197, reset 0x00 (OTG A / Host Mode) (see page 1013)															
					AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT				
USBRXCSRH10, type R/W, offset 0x1A7, reset 0x00 (OTG A / Host Mode) (see page 1013)															
					AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT				
USBRXCSRH11, type R/W, offset 0x1B7, reset 0x00 (OTG A / Host Mode) (see page 1013)															
					AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT				
USBRXCSRH12, type R/W, offset 0x1C7, reset 0x00 (OTG A / Host Mode) (see page 1013)															
					AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT				
USBRXCSRH13, type R/W, offset 0x1D7, reset 0x00 (OTG A / Host Mode) (see page 1013)															
					AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT				
USBRXCSRH14, type R/W, offset 0x1E7, reset 0x00 (OTG A / Host Mode) (see page 1013)															
					AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT				
USBRXCSRH15, type R/W, offset 0x1F7, reset 0x00 (OTG A / Host Mode) (see page 1013)															
					AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT				
USBRXCSRH1, type R/W, offset 0x117, reset 0x00 (OTG B / Device Mode) (see page 1013)															
					AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD						
USBRXCSRH2, type R/W, offset 0x127, reset 0x00 (OTG B / Device Mode) (see page 1013)															
					AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD						
USBRXCSRH3, type R/W, offset 0x137, reset 0x00 (OTG B / Device Mode) (see page 1013)															
					AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD						
USBRXCSRH4, type R/W, offset 0x147, reset 0x00 (OTG B / Device Mode) (see page 1013)															
					AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD						
USBRXCSRH5, type R/W, offset 0x157, reset 0x00 (OTG B / Device Mode) (see page 1013)															
					AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD						
USBRXCSRH6, type R/W, offset 0x167, reset 0x00 (OTG B / Device Mode) (see page 1013)															
					AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD						
USBRXCSRH7, type R/W, offset 0x177, reset 0x00 (OTG B / Device Mode) (see page 1013)															
					AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD						
USBRXCSRH8, type R/W, offset 0x187, reset 0x00 (OTG B / Device Mode) (see page 1013)															
					AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD						

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
USBRXCSRH9, type R/W, offset 0x197, reset 0x00 (OTG B / Device Mode) (see page 1013)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH10, type R/W, offset 0x1A7, reset 0x00 (OTG B / Device Mode) (see page 1013)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH11, type R/W, offset 0x1B7, reset 0x00 (OTG B / Device Mode) (see page 1013)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH12, type R/W, offset 0x1C7, reset 0x00 (OTG B / Device Mode) (see page 1013)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH13, type R/W, offset 0x1D7, reset 0x00 (OTG B / Device Mode) (see page 1013)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH14, type R/W, offset 0x1E7, reset 0x00 (OTG B / Device Mode) (see page 1013)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCSRH15, type R/W, offset 0x1F7, reset 0x00 (OTG B / Device Mode) (see page 1013)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			
USBRXCOUNT1, type RO, offset 0x118, reset 0x0000 (see page 1018)															
								COUNT							
USBRXCOUNT2, type RO, offset 0x128, reset 0x0000 (see page 1018)															
								COUNT							
USBRXCOUNT3, type RO, offset 0x138, reset 0x0000 (see page 1018)															
								COUNT							
USBRXCOUNT4, type RO, offset 0x148, reset 0x0000 (see page 1018)															
								COUNT							
USBRXCOUNT5, type RO, offset 0x158, reset 0x0000 (see page 1018)															
								COUNT							
USBRXCOUNT6, type RO, offset 0x168, reset 0x0000 (see page 1018)															
								COUNT							
USBRXCOUNT7, type RO, offset 0x178, reset 0x0000 (see page 1018)															
								COUNT							
USBRXCOUNT8, type RO, offset 0x188, reset 0x0000 (see page 1018)															
								COUNT							
USBRXCOUNT9, type RO, offset 0x198, reset 0x0000 (see page 1018)															
								COUNT							
USBRXCOUNT10, type RO, offset 0x1A8, reset 0x0000 (see page 1018)															
								COUNT							
USBRXCOUNT11, type RO, offset 0x1B8, reset 0x0000 (see page 1018)															
								COUNT							
USBRXCOUNT12, type RO, offset 0x1C8, reset 0x0000 (see page 1018)															
								COUNT							
USBRXCOUNT13, type RO, offset 0x1D8, reset 0x0000 (see page 1018)															
								COUNT							
USBRXCOUNT14, type RO, offset 0x1E8, reset 0x0000 (see page 1018)															
								COUNT							
USBRXCOUNT15, type RO, offset 0x1F8, reset 0x0000 (see page 1018)															
								COUNT							
USBTYPE1, type R/W, offset 0x11A, reset 0x00 (see page 1020)															
								SPEED	PROTO		TEP				
USBTYPE2, type R/W, offset 0x12A, reset 0x00 (see page 1020)															
								SPEED	PROTO		TEP				

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
USBTXTYPE3, type R/W, offset 0x13A, reset 0x00 (see page 1020)															
								SPEED	PROTO			TEP			
USBTXTYPE4, type R/W, offset 0x14A, reset 0x00 (see page 1020)															
								SPEED	PROTO			TEP			
USBTXTYPE5, type R/W, offset 0x15A, reset 0x00 (see page 1020)															
								SPEED	PROTO			TEP			
USBTXTYPE6, type R/W, offset 0x16A, reset 0x00 (see page 1020)															
								SPEED	PROTO			TEP			
USBTXTYPE7, type R/W, offset 0x17A, reset 0x00 (see page 1020)															
								SPEED	PROTO			TEP			
USBTXTYPE8, type R/W, offset 0x18A, reset 0x00 (see page 1020)															
								SPEED	PROTO			TEP			
USBTXTYPE9, type R/W, offset 0x19A, reset 0x00 (see page 1020)															
								SPEED	PROTO			TEP			
USBTXTYPE10, type R/W, offset 0x1AA, reset 0x00 (see page 1020)															
								SPEED	PROTO			TEP			
USBTXTYPE11, type R/W, offset 0x1BA, reset 0x00 (see page 1020)															
								SPEED	PROTO			TEP			
USBTXTYPE12, type R/W, offset 0x1CA, reset 0x00 (see page 1020)															
								SPEED	PROTO			TEP			
USBTXTYPE13, type R/W, offset 0x1DA, reset 0x00 (see page 1020)															
								SPEED	PROTO			TEP			
USBTXTYPE14, type R/W, offset 0x1EA, reset 0x00 (see page 1020)															
								SPEED	PROTO			TEP			
USBTXTYPE15, type R/W, offset 0x1FA, reset 0x00 (see page 1020)															
								SPEED	PROTO			TEP			
USBTXINTERVAL1, type R/W, offset 0x11B, reset 0x00 (see page 1022)															
													TXPOLL / NAKLMT		
USBTXINTERVAL2, type R/W, offset 0x12B, reset 0x00 (see page 1022)															
													TXPOLL / NAKLMT		
USBTXINTERVAL3, type R/W, offset 0x13B, reset 0x00 (see page 1022)															
													TXPOLL / NAKLMT		
USBTXINTERVAL4, type R/W, offset 0x14B, reset 0x00 (see page 1022)															
													TXPOLL / NAKLMT		
USBTXINTERVAL5, type R/W, offset 0x15B, reset 0x00 (see page 1022)															
													TXPOLL / NAKLMT		
USBTXINTERVAL6, type R/W, offset 0x16B, reset 0x00 (see page 1022)															
													TXPOLL / NAKLMT		
USBTXINTERVAL7, type R/W, offset 0x17B, reset 0x00 (see page 1022)															
													TXPOLL / NAKLMT		
USBTXINTERVAL8, type R/W, offset 0x18B, reset 0x00 (see page 1022)															
													TXPOLL / NAKLMT		
USBTXINTERVAL9, type R/W, offset 0x19B, reset 0x00 (see page 1022)															
													TXPOLL / NAKLMT		
USBTXINTERVAL10, type R/W, offset 0x1AB, reset 0x00 (see page 1022)															
													TXPOLL / NAKLMT		
USBTXINTERVAL11, type R/W, offset 0x1BB, reset 0x00 (see page 1022)															
													TXPOLL / NAKLMT		
USBTXINTERVAL12, type R/W, offset 0x1CB, reset 0x00 (see page 1022)															
													TXPOLL / NAKLMT		
USBTXINTERVAL13, type R/W, offset 0x1DB, reset 0x00 (see page 1022)															
													TXPOLL / 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
USBTXINTERVAL14, type R/W, offset 0x1EB, reset 0x00 (see page 1022)															
															TXPOLL / NAKLMT
USBTXINTERVAL15, type R/W, offset 0x1FB, reset 0x00 (see page 1022)															
															TXPOLL / NAKLMT
USBRXTYPE1, type R/W, offset 0x11C, reset 0x00 (see page 1024)															
															SPEED PROTO TEP
USBRXTYPE2, type R/W, offset 0x12C, reset 0x00 (see page 1024)															
															SPEED PROTO TEP
USBRXTYPE3, type R/W, offset 0x13C, reset 0x00 (see page 1024)															
															SPEED PROTO TEP
USBRXTYPE4, type R/W, offset 0x14C, reset 0x00 (see page 1024)															
															SPEED PROTO TEP
USBRXTYPE5, type R/W, offset 0x15C, reset 0x00 (see page 1024)															
															SPEED PROTO TEP
USBRXTYPE6, type R/W, offset 0x16C, reset 0x00 (see page 1024)															
															SPEED PROTO TEP
USBRXTYPE7, type R/W, offset 0x17C, reset 0x00 (see page 1024)															
															SPEED PROTO TEP
USBRXTYPE8, type R/W, offset 0x18C, reset 0x00 (see page 1024)															
															SPEED PROTO TEP
USBRXTYPE9, type R/W, offset 0x19C, reset 0x00 (see page 1024)															
															SPEED PROTO TEP
USBRXTYPE10, type R/W, offset 0x1AC, reset 0x00 (see page 1024)															
															SPEED PROTO TEP
USBRXTYPE11, type R/W, offset 0x1BC, reset 0x00 (see page 1024)															
															SPEED PROTO TEP
USBRXTYPE12, type R/W, offset 0x1CC, reset 0x00 (see page 1024)															
															SPEED PROTO TEP
USBRXTYPE13, type R/W, offset 0x1DC, reset 0x00 (see page 1024)															
															SPEED PROTO TEP
USBRXTYPE14, type R/W, offset 0x1EC, reset 0x00 (see page 1024)															
															SPEED PROTO TEP
USBRXTYPE15, type R/W, offset 0x1FC, reset 0x00 (see page 1024)															
															SPEED PROTO TEP
USBRXINTERVAL1, type R/W, offset 0x11D, reset 0x00 (see page 1026)															
															TXPOLL / NAKLMT
USBRXINTERVAL2, type R/W, offset 0x12D, reset 0x00 (see page 1026)															
															TXPOLL / NAKLMT
USBRXINTERVAL3, type R/W, offset 0x13D, reset 0x00 (see page 1026)															
															TXPOLL / NAKLMT
USBRXINTERVAL4, type R/W, offset 0x14D, reset 0x00 (see page 1026)															
															TXPOLL / NAKLMT
USBRXINTERVAL5, type R/W, offset 0x15D, reset 0x00 (see page 1026)															
															TXPOLL / NAKLMT
USBRXINTERVAL6, type R/W, offset 0x16D, reset 0x00 (see page 1026)															
															TXPOLL / NAKLMT
USBRXINTERVAL7, type R/W, offset 0x17D, reset 0x00 (see page 1026)															
															TXPOLL / NAKLMT
USBRXINTERVAL8, type R/W, offset 0x18D, reset 0x00 (see page 1026)															
															TXPOLL / NAKLMT
USBRXINTERVAL9, type R/W, offset 0x19D, reset 0x00 (see page 1026)															
															TXPOLL / 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
ACREFCTL , type R/W, offset 0x010, reset 0x0000.0000 (see page 1061)															
					EN	RNG						VREF			
ACSTAT0 , type RO, offset 0x020, reset 0x0000.0000 (see page 1062)															
												OVAL			
ACSTAT1 , type RO, offset 0x040, reset 0x0000.0000 (see page 1062)															
												OVAL			
ACCTL0 , type R/W, offset 0x024, reset 0x0000.0000 (see page 1063)															
		TOEN	ASRCP		TSLVAL	TSEN	ISLVAL		ISEN	CINV					
ACCTL1 , type R/W, offset 0x044, reset 0x0000.0000 (see page 1063)															
		TOEN	ASRCP		TSLVAL	TSEN	ISLVAL		ISEN	CINV					
Pulse Width Modulator (PWM)															
Base 0x4002.8000															
PWMCTL , type R/W, offset 0x000, reset 0x0000.0000 (see page 1079)															
												GLOBALSYNC2	GLOBALSYNC1	GLOBALSYNC0	
PWMSYNC , type R/W, offset 0x004, reset 0x0000.0000 (see page 1080)															
												SYNC2	SYNC1	SYNC0	
PWMENABLE , type R/W, offset 0x008, reset 0x0000.0000 (see page 1081)															
									PWM5EN	PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN	
PWMINVERT , type R/W, offset 0x00C, reset 0x0000.0000 (see page 1083)									PWM5INV	PWM4INV	PWM3INV	PWM2INV	PWM1INV	PWM0INV	
PWMFAULT , type R/W, offset 0x010, reset 0x0000.0000 (see page 1085)									FAULT5	FAULT4	FAULT3	FAULT2	FAULT1	FAULT0	
PWMINTEN , type R/W, offset 0x014, reset 0x0000.0000 (see page 1087)											INTFAULT3	INTFAULT2	INTFAULT1	INTFAULT0	
											INTPWM2	INTPWM1	INTPWM0		
PWMRIS , type RO, offset 0x018, reset 0x0000.0000 (see page 1089)											INTFAULT3	INTFAULT2	INTFAULT1	INTFAULT0	
											INTPWM2	INTPWM1	INTPWM0		
PWMISC , type R/W1C, offset 0x01C, reset 0x0000.0000 (see page 1091)											INTFAULT3	INTFAULT2	INTFAULT1	INTFAULT0	
											INTPWM2	INTPWM1	INTPWM0		
PWMSTATUS , type RO, offset 0x020, reset 0x0000.0000 (see page 1093)											FAULT3	FAULT2	FAULT1	FAULT0	
PWMFAULTVAL , type R/W, offset 0x024, reset 0x0000.0000 (see page 1095)									PWM5	PWM4	PWM3	PWM2	PWM1	PWM0	
PWMENUPD , type R/W, offset 0x028, reset 0x0000.0000 (see page 1097)									ENUPD5	ENUPD4	ENUPD3	ENUPD2	ENUPD1	ENUPD0	
PWM0CTL , type R/W, offset 0x040, reset 0x0000.0000 (see page 1100)											LATCH	MINFLTPER	FLTSRC		
DBFALLUPD	DBRISEUPD	DBCTLU	GENBUPD	GENAUPD	CMPBUPD	CMPAUPD	LOADUPD	DEBUG	MODE	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
PWM1CTL , type R/W, offset 0x080, reset 0x0000.0000 (see page 1100)															
DBFALLUPD	DBRISEUPD	DBCTLUPD	GENBUPD	GENAUPD	CMPBUPD	CMPAUPD	LOADUPD	LATCH	MINFLTPER	FLTSRC	DEBUG	MODE	ENABLE		
PWM2CTL , type R/W, offset 0x0C0, reset 0x0000.0000 (see page 1100)															
DBFALLUPD	DBRISEUPD	DBCTLUPD	GENBUPD	GENAUPD	CMPBUPD	CMPAUPD	LOADUPD	LATCH	MINFLTPER	FLTSRC	DEBUG	MODE	ENABLE		
PWM0INTEN , type R/W, offset 0x044, reset 0x0000.0000 (see page 1105)															
TRCMPBD	TRCMPBU	TRCPAD	TRCPAU	TRCNTLOAD	TRCNZERO			INTCMPBD	INTCMPBU	INTCPAD	INTCPAU	INTCNTLOAD	INTCNZERO		
PWM1INTEN , type R/W, offset 0x084, reset 0x0000.0000 (see page 1105)															
TRCMPBD	TRCMPBU	TRCPAD	TRCPAU	TRCNTLOAD	TRCNZERO			INTCMPBD	INTCMPBU	INTCPAD	INTCPAU	INTCNTLOAD	INTCNZERO		
PWM2INTEN , type R/W, offset 0x0C4, reset 0x0000.0000 (see page 1105)															
TRCMPBD	TRCMPBU	TRCPAD	TRCPAU	TRCNTLOAD	TRCNZERO			INTCMPBD	INTCMPBU	INTCPAD	INTCPAU	INTCNTLOAD	INTCNZERO		
PWM0RIS , type RO, offset 0x048, reset 0x0000.0000 (see page 1108)															
								INTCMPBD	INTCMPBU	INTCPAD	INTCPAU	INTCNTLOAD	INTCNZERO		
PWM1RIS , type RO, offset 0x088, reset 0x0000.0000 (see page 1108)															
								INTCMPBD	INTCMPBU	INTCPAD	INTCPAU	INTCNTLOAD	INTCNZERO		
PWM2RIS , type RO, offset 0x0C8, reset 0x0000.0000 (see page 1108)															
								INTCMPBD	INTCMPBU	INTCPAD	INTCPAU	INTCNTLOAD	INTCNZERO		
PWM0ISC , type R/W1C, offset 0x04C, reset 0x0000.0000 (see page 1110)															
								INTCMPBD	INTCMPBU	INTCPAD	INTCPAU	INTCNTLOAD	INTCNZERO		
PWM1ISC , type R/W1C, offset 0x08C, reset 0x0000.0000 (see page 1110)															
								INTCMPBD	INTCMPBU	INTCPAD	INTCPAU	INTCNTLOAD	INTCNZERO		
PWM2ISC , type R/W1C, offset 0x0CC, reset 0x0000.0000 (see page 1110)															
								INTCMPBD	INTCMPBU	INTCPAD	INTCPAU	INTCNTLOAD	INTCNZERO		
PWM0LOAD , type R/W, offset 0x050, reset 0x0000.0000 (see page 1112)															
								LOAD							
PWM1LOAD , type R/W, offset 0x090, reset 0x0000.0000 (see page 1112)															
								LOAD							
PWM2LOAD , type R/W, offset 0x0D0, reset 0x0000.0000 (see page 1112)															
								LOAD							
PWM0COUNT , type RO, offset 0x054, reset 0x0000.0000 (see page 1113)															
								COUNT							
PWM1COUNT , type RO, offset 0x094, reset 0x0000.0000 (see page 1113)															
								COUNT							
PWM2COUNT , type RO, offset 0x0D4, reset 0x0000.0000 (see page 1113)															
								COUNT							

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												
PWM0CMPA, type R/W, offset 0x058, reset 0x0000.0000 (see page 1114)																											
COMPA																											
PWM1CMPA, type R/W, offset 0x098, reset 0x0000.0000 (see page 1114)																											
COMPA																											
PWM2CMPA, type R/W, offset 0x0D8, reset 0x0000.0000 (see page 1114)																											
COMPA																											
PWM0CMPB, type R/W, offset 0x05C, reset 0x0000.0000 (see page 1115)																											
COMPB																											
PWM1CMPB, type R/W, offset 0x09C, reset 0x0000.0000 (see page 1115)																											
COMPB																											
PWM2CMPB, type R/W, offset 0x0DC, reset 0x0000.0000 (see page 1115)																											
COMPB																											
PWM0GENA, type R/W, offset 0x060, reset 0x0000.0000 (see page 1116)																											
ACTCMPBD				ACTCMPBU				ACTCMPAD				ACTCMPPAU															
PWM1GENA, type R/W, offset 0x0A0, reset 0x0000.0000 (see page 1116)				ACTCMPBD				ACTCMPBU				ACTCMPPAD															
ACTLOAD				ACTZERO				ACTLOAD				ACTZERO															
PWM2GENA, type R/W, offset 0x0E0, reset 0x0000.0000 (see page 1116)																											
ACTCMPBD				ACTCMPBU				ACTCMPPAD				ACTCMPPAU															
ACTLOAD				ACTZERO				ACTLOAD				ACTZERO															
PWM0GENB, type R/W, offset 0x064, reset 0x0000.0000 (see page 1119)																											
ACTCMPBD				ACTCMPBU				ACTCMPPAD				ACTCMPPAU															
ACTLOAD				ACTZERO				ACTLOAD				ACTZERO															
PWM1GENB, type R/W, offset 0x0A4, reset 0x0000.0000 (see page 1119)																											
ACTCMPBD				ACTCMPBU				ACTCMPPAD				ACTCMPPAU															
ACTLOAD				ACTZERO				ACTLOAD				ACTZERO															
PWM2GENB, type R/W, offset 0x0E4, reset 0x0000.0000 (see page 1119)																											
ACTCMPBD				ACTCMPBU				ACTCMPPAD				ACTCMPPAU															
ACTLOAD				ACTZERO				ACTLOAD				ACTZERO															
PWM0DBCTL, type R/W, offset 0x068, reset 0x0000.0000 (see page 1122)																											
ENABLE																											
PWM1DBCTL, type R/W, offset 0xA8, reset 0x0000.0000 (see page 1122)																											
ENABLE																											
PWM2DBCTL, type R/W, offset 0xE8, reset 0x0000.0000 (see page 1122)																											
ENABLE																											
PWM0DBRISE, type R/W, offset 0x06C, reset 0x0000.0000 (see page 1123)																											
RISEDELAY																											
PWM1DBRISE, type R/W, offset 0x0AC, reset 0x0000.0000 (see page 1123)																											
RISEDELAY																											

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
PWM2DBRISE , type R/W, offset 0x0EC, reset 0x0000.0000 (see page 1123)															
RISEDELAY															
PWM0DBFALL , type R/W, offset 0x070, reset 0x0000.0000 (see page 1124)															
FALLDELAY															
PWM1DBFALL , type R/W, offset 0x0B0, reset 0x0000.0000 (see page 1124)															
FALLDELAY															
PWM2DBFALL , type R/W, offset 0x0F0, reset 0x0000.0000 (see page 1124)															
FALLDELAY															
PWM0FLTSRC0 , type R/W, offset 0x074, reset 0x0000.0000 (see page 1125)															
FAULT3 FAULT2 FAULT1 FAULT0															
PWM1FLTSRC0 , type R/W, offset 0x0B4, reset 0x0000.0000 (see page 1125)															
FAULT3 FAULT2 FAULT1 FAULT0															
PWM2FLTSRC0 , type R/W, offset 0x0F4, reset 0x0000.0000 (see page 1125)															
FAULT3 FAULT2 FAULT1 FAULT0															
PWM0FLTSRC1 , type R/W, offset 0x078, reset 0x0000.0000 (see page 1127)															
DCMP7 DCMP6 DCMP5 DCMP4 DCMP3 DCMP2 DCMP1 DCMP0															
PWM1FLTSRC1 , type R/W, offset 0x0B8, reset 0x0000.0000 (see page 1127)															
DCMP7 DCMP6 DCMP5 DCMP4 DCMP3 DCMP2 DCMP1 DCMP0															
PWM2FLTSRC1 , type R/W, offset 0x0F8, reset 0x0000.0000 (see page 1127)															
DCMP7 DCMP6 DCMP5 DCMP4 DCMP3 DCMP2 DCMP1 DCMP0															
PWM0MINFLTPER , type R/W, offset 0x07C, reset 0x0000.0000 (see page 1130)															
MFP															
PWM1MINFLTPER , type R/W, offset 0x0BC, reset 0x0000.0000 (see page 1130)															
MFP															
PWM2MINFLTPER , type R/W, offset 0x0FC, reset 0x0000.0000 (see page 1130)															
MFP															
PWM0FLTSEN , type R/W, offset 0x800, reset 0x0000.0000 (see page 1131)															
FAULT3 FAULT2 FAULT1 FAULT0															
PWM1FLTSEN , type R/W, offset 0x880, reset 0x0000.0000 (see page 1131)															
FAULT3 FAULT2 FAULT1 FAULT0															
PWM2FLTSEN , type R/W, offset 0x900, reset 0x0000.0000 (see page 1131)															
FAULT3 FAULT2 FAULT1 FAULT0															
PWM3FLTSEN , type R/W, offset 0x980, reset 0x0000.0000 (see page 1131)															
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				
PWM0FLTSTAT0 , type -, offset 0x804, reset 0x0000.0000 (see page 1132)																			
														FAULT3	FAULT2	FAULT1	FAULT0		
PWM1FLTSTAT0 , type -, offset 0x884, reset 0x0000.0000 (see page 1132)																			
														FAULT3	FAULT2	FAULT1	FAULT0		
PWM2FLTSTAT0 , type -, offset 0x904, reset 0x0000.0000 (see page 1132)																			
														FAULT3	FAULT2	FAULT1	FAULT0		
PWM0FLTSTAT1 , type -, offset 0x808, reset 0x0000.0000 (see page 1134)																			
								DCMP7	DCMP6	DCMP5	DCMP4	DCMP3	DCMP2	DCMP1	DCMP0				
PWM1FLTSTAT1 , type -, offset 0x888, reset 0x0000.0000 (see page 1134)								DCMP7	DCMP6	DCMP5	DCMP4	DCMP3	DCMP2	DCMP1	DCMP0				
PWM2FLTSTAT1 , type -, offset 0x908, reset 0x0000.0000 (see page 1134)								DCMP7	DCMP6	DCMP5	DCMP4	DCMP3	DCMP2	DCMP1	DCMP0				
Quadrature Encoder Interface (QEI)																			
QEIO base: 0x4002.C000								FILTCNT											
QEII base: 0x4002.D000								FILTEN	STALLEN	INVI	INVB	INVA	VELDIV	VELEN	RESMODE	CAPMODE	SIGMODE	SWAP	ENABLE
QEICTL , type R/W, offset 0x000, reset 0x0000.0000 (see page 1144)																			
QEISTAT, type RO, offset 0x004, reset 0x0000.0000 (see page 1147)								DIRECTION ERROR											
QEIPOS , type R/W, offset 0x008, reset 0x0000.0000 (see page 1148)																			
POSITION								POSITION											
QEIMAXPOS , type R/W, offset 0x00C, reset 0x0000.0000 (see page 1149)																			
MAXPOS								MAXPOS											
QEILOAD , type R/W, offset 0x010, reset 0x0000.0000 (see page 1150)																			
LOAD								LOAD											
QEITIME , type RO, offset 0x014, reset 0x0000.0000 (see page 1151)																			
TIME								TIME											
QEICOUNT , type RO, offset 0x018, reset 0x0000.0000 (see page 1152)																			
COUNT								COUNT											
QEISPEED , type RO, offset 0x01C, reset 0x0000.0000 (see page 1153)																			
SPEED								SPEED											
QEINTEN , type R/W, offset 0x020, reset 0x0000.0000 (see page 1154)																			
								INTERROR INTDIR INTTIMER INTINDEX											
QEIRIS , type RO, offset 0x024, reset 0x0000.0000 (see page 1156)																			
								INTERROR INTDIR INTTIMER INTINDEX											

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
QEIIISC, type R/W1C, offset 0x028, reset 0x0000.0000 (see page 1158)															

B Ordering and Contact Information

B.1 Ordering Information

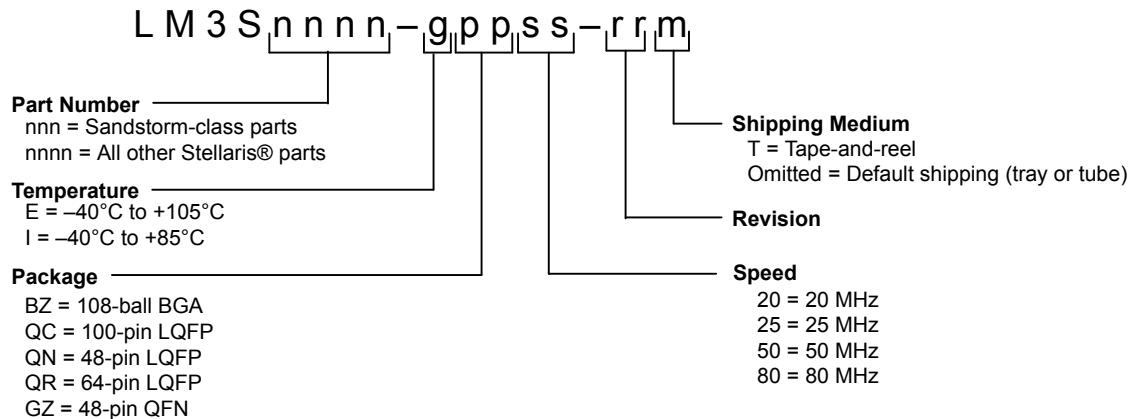


Table B-1. Part Ordering Information

Orderable Part Number	Description
LM3S9L97-IQC80-C3	Stellaris® LM3S9L97 Microcontroller Industrial Temperature 100-pin LQFP
LM3S9L97-IBZ80-C3	Stellaris LM3S9L97 Microcontroller Industrial Temperature 108-ball BGA
LM3S9L97-IQC80-C3T	Stellaris LM3S9L97 Microcontroller Industrial Temperature 100-pin LQFP Tape-and-reel
LM3S9L97-IBZ80-C3T	Stellaris LM3S9L97 Microcontroller Industrial Temperature 108-ball BGA Tape-and-reel

B.2 Part Markings

The Stellaris microcontrollers are marked with an identifying number. This code contains the following information:

- The first line indicates the part number. In the example figure below, this is the LM3S9B90.
- In the second line, the first seven characters indicate the temperature, package, speed, and revision. In the example below, this is an Industrial temperature (I), 100-pin LQFP package (QC), 80-MHz (80), revision C0 (C0) device.
- The remaining characters contain internal tracking numbers.



B.3 Kits

The Stellaris Family provides the hardware and software tools that engineers need to begin development quickly.

- Reference Design Kits accelerate product development by providing ready-to-run hardware and comprehensive documentation including hardware design files
- Evaluation Kits provide a low-cost and effective means of evaluating Stellaris microcontrollers before purchase
- Development Kits provide you with all the tools you need to develop and prototype embedded applications right out of the box

See the website at www.ti.com/stellaris for the latest tools available, or ask your distributor.

B.4 Support Information

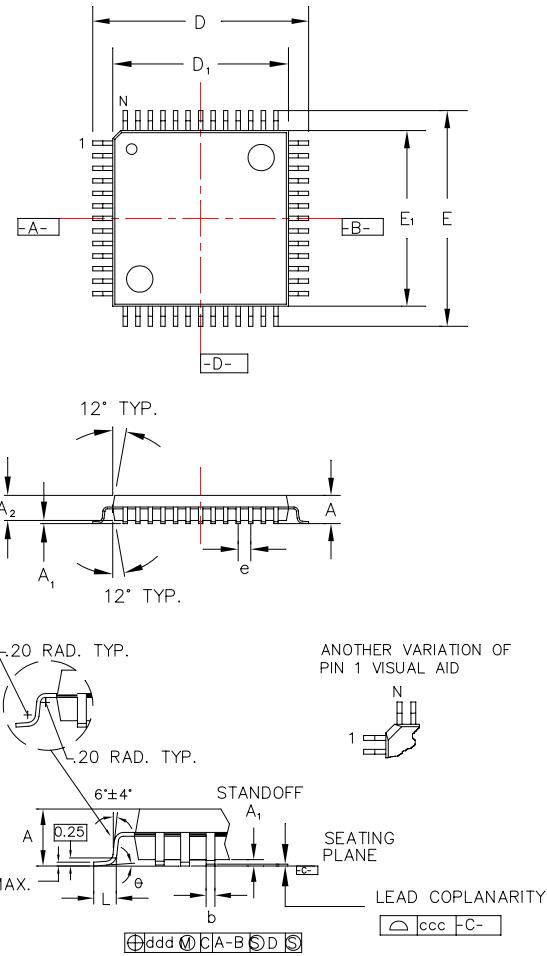
For support on Stellaris products, contact the TI Worldwide Product Information Center nearest you:
<http://www-k.ext.ti.com/sc/technical-support/product-information-centers.htm>.

C Package Information

C.1 100-Pin LQFP Package

C.1.1 Package Dimensions

Figure C-1. 100-Pin LQFP Package Dimensions



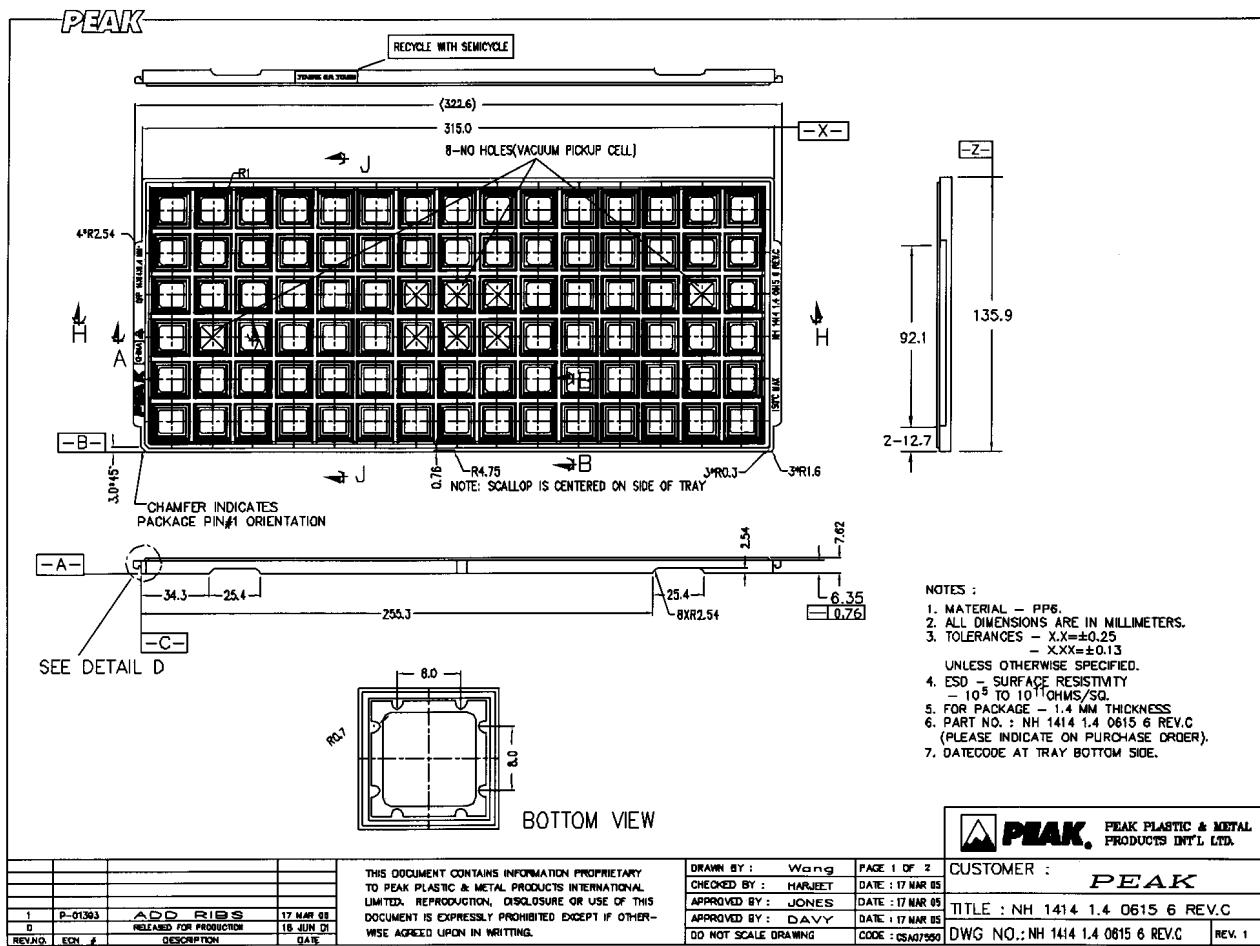
Note: The following notes apply to the package drawing.

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	
A	Max.	1.60
A ₁	-	0.05 Min./0.15 Max.
A ₂	±0.05	1.40
D	±0.20	16.00
D ₁	±0.05	14.00
E	±0.20	16.00
E ₁	±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

C.1.2 Tray Dimensions

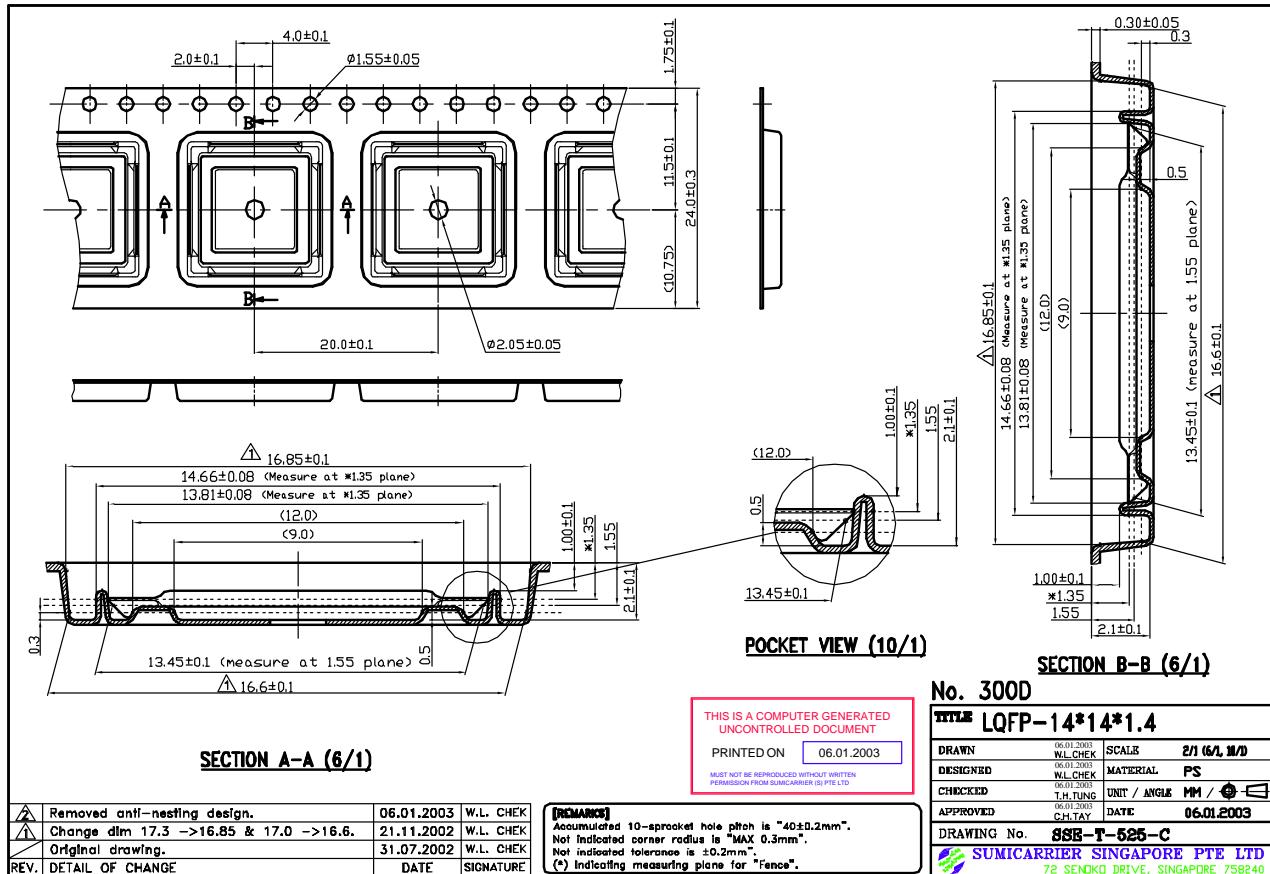
Figure C-2. 100-Pin LQFP Tray Dimensions



C.1.3 Tape and Reel Dimensions

Note: In the figure that follows, pin 1 is located in the top right corner of the device.

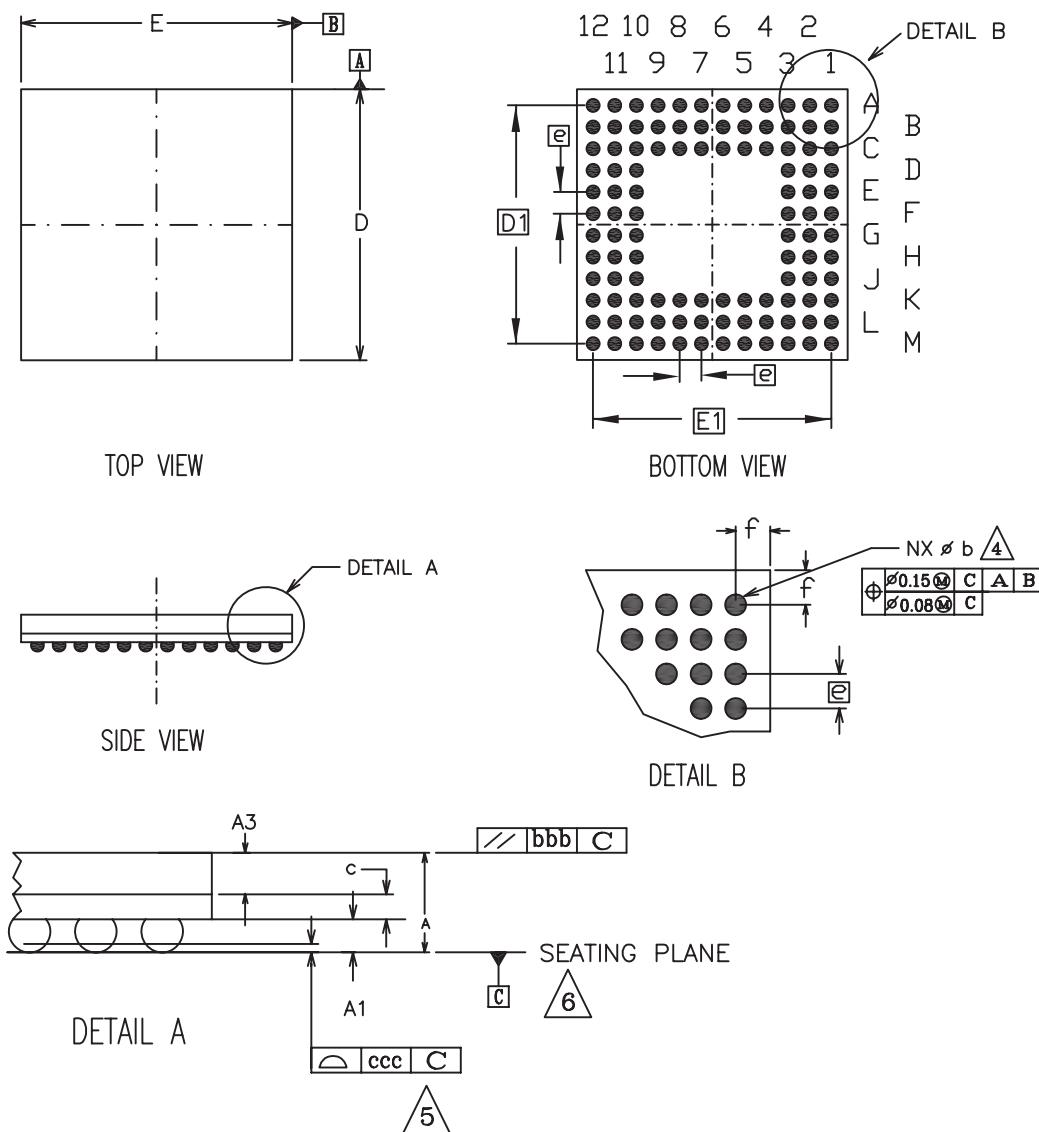
Figure C-3. 100-Pin LQFP Tape and Reel Dimensions



C.2 108-Ball BGA Package

C.2.1 Package Dimensions

Figure C-4. 108-Ball BGA Package Dimensions



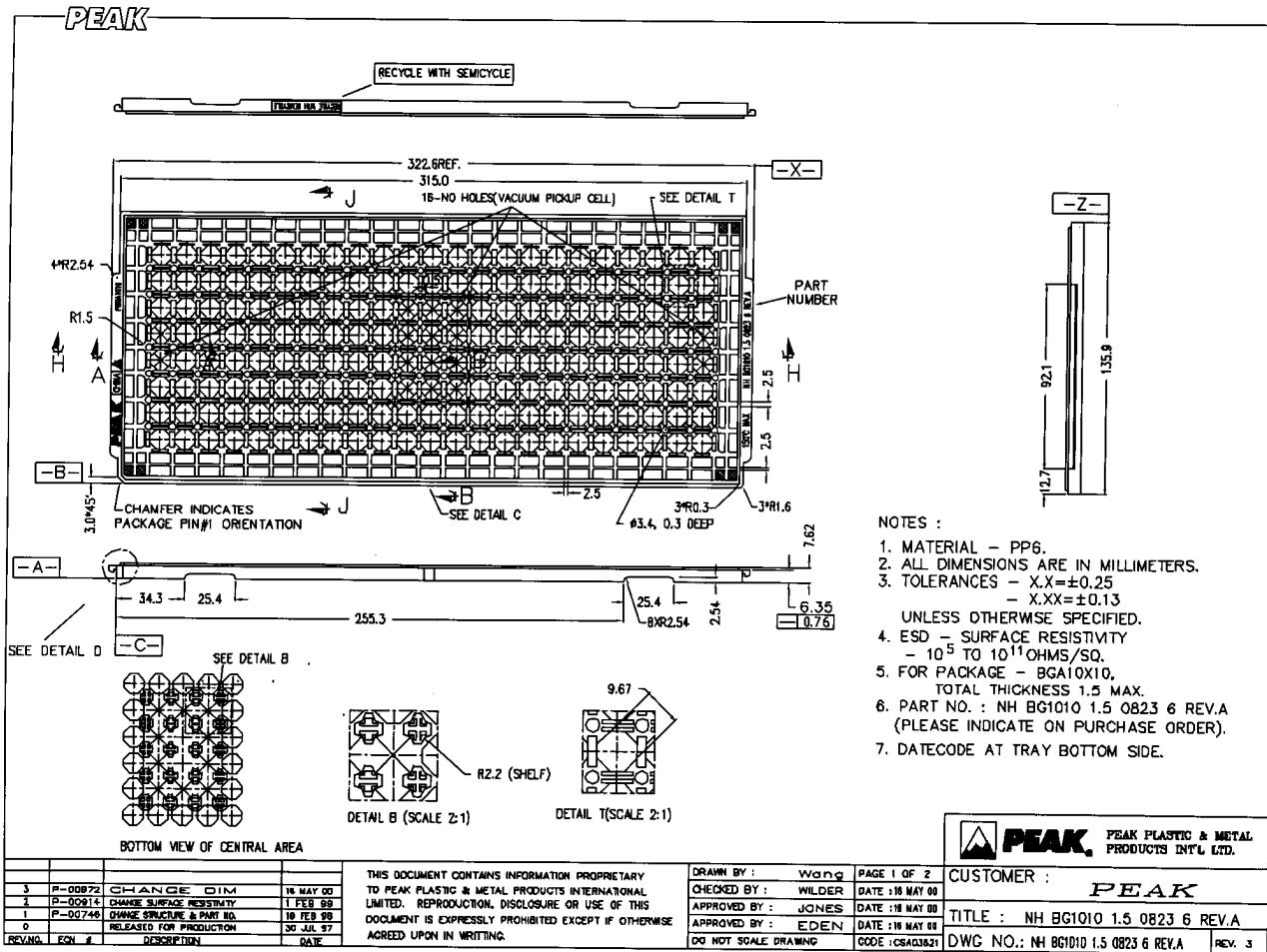
Note: The following notes apply to the package drawing.

1. ALL DIMENSIONS ARE IN MILLIMETERS.
 2. 'e' REPRESENTS THE BASIC SOLDER BALL GRID PITCH.
 3. 'M' REPRESENTS THE BASIC SOLDER BALL MATRIX SIZE.
AND SYMBOL 'N' IS THE NUMBER OF BALLS AFTER DEPOPULATING.
- 'b' IS MEASURABLE AT THE MAXIMUM SOLDER BALL DIAMETER AFTER REFLOW PARALLEL TO PRIMARY DATUM .
- DIMENSION 'ccc' IS MEASURED PARALLEL TO PRIMARY DATUM .
- PRIMARY DATUM AND SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
7. PACKAGE SURFACE SHALL BE MATTE FINISH CHARMILLES 24 TO 27.
 8. SUBSTRATE MATERIAL BASE IS BT RESIN.
 9. THE OVERALL PACKAGE THICKNESS "A" ALREADY CONSIDERS COLLAPSE BALLS
 10. DIMENSIONING AND TOLERANCING PER ASME Y14.5M 1994.
- EXCEPT DIMENSION b.

Symbols	MIN	NOM	MAX
A	1.22	1.36	1.50
A1	0.29	0.34	0.39
A3	0.65	0.70	0.75
c	0.28	0.32	0.36
D	9.85	10.00	10.15
D1		8.80 BSC	
E	9.85	10.00	10.15
E1		8.80 BSC	
b	0.43	0.48	0.53
bbb		.20	
ddd		.12	
e		0.80 BSC	
f	-	0.60	-
M		12	
n		108	
REF: JEDEC MO-219F			

C.2.2 Tray Dimensions

Figure C-5. 108-Ball BGA Tray Dimensions



C.2.3 Tape and Reel Dimensions

Figure C-6. 108-Ball BGA Tape and Reel Dimensions

