



LUMINARY MICRO™

LM3S8938 Microcontroller

DATA SHEET

Legal Disclaimers and Trademark Information

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

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

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

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

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



LUMINARY MICRO™



Table of Contents

About This Document	20
Audience	20
About This Manual	20
Related Documents	20
Documentation Conventions	20
1 Overview	22
1.1 Product Features	22
1.2 Target Applications	27
1.3 High-Level Block Diagram	27
1.4 Functional Overview	28
1.4.1 ARM Cortex™-M3	29
1.4.2 Motor Control Peripherals	29
1.4.3 Serial Communications Peripherals	30
1.4.4 System Peripherals	31
1.4.5 Memory Peripherals	32
1.4.6 Additional Features	33
1.4.7 Hardware Details	33
2 Cortex-M3 Core	35
2.1 Block Diagram	36
2.2 Functional Description	36
2.2.1 Serial Wire and JTAG Debug	36
2.2.2 Embedded Trace Macrocell (ETM)	37
2.2.3 Trace Port Interface Unit (TPIU)	37
2.2.4 ROM Table	37
2.2.5 Memory Protection Unit (MPU)	37
2.2.6 Nested Vectored Interrupt Controller (NVIC)	37
3 Memory Map	41
4 Interrupts	43
5 JTAG	46
5.1 Block Diagram	47
5.2 Functional Description	47
5.2.1 JTAG Interface Pins	48
5.2.2 JTAG TAP Controller	49
5.2.3 Shift Registers	50
5.2.4 Operational Considerations	50
5.3 Initialization and Configuration	53
5.4 Register Descriptions	53
5.4.1 Instruction Register (IR)	53
5.4.2 Data Registers	55
6 System Control	57
6.1 Functional Description	57
6.1.1 Device Identification	57
6.1.2 Reset Control	57
6.1.3 Power Control	60

6.1.4	Clock Control	60
6.1.5	System Control	62
6.2	Initialization and Configuration	63
6.3	Register Map	63
6.4	Register Descriptions	64
7	Hibernation Module	113
7.1	Block Diagram	114
7.2	Functional Description	114
7.2.1	Register Access Timing	114
7.2.2	Clock Source	115
7.2.3	Battery Management	115
7.2.4	Real-Time Clock	115
7.2.5	Non-Volatile Memory	116
7.2.6	Power Control	116
7.2.7	Interrupts and Status	116
7.3	Initialization and Configuration	116
7.3.1	Initialization	117
7.3.2	RTC Match Functionality (No Hibernation)	117
7.3.3	RTC Match/Wake-Up from Hibernation	117
7.3.4	External Wake-Up from Hibernation	117
7.3.5	RTC/External Wake-Up from Hibernation	118
7.4	Register Map	118
7.5	Register Descriptions	118
8	Internal Memory	131
8.1	Block Diagram	131
8.2	Functional Description	131
8.2.1	SRAM Memory	131
8.2.2	Flash Memory	132
8.3	Flash Memory Initialization and Configuration	133
8.3.1	Flash Programming	133
8.3.2	Nonvolatile Register Programming	134
8.4	Register Map	134
8.5	Flash Control Offset	135
8.6	System Control Offset	142
9	GPIO	155
9.1	Function Description	155
9.1.1	Data Control	155
9.1.2	Interrupt Control	156
9.1.3	Mode Control	157
9.1.4	Commit Control	157
9.1.5	Pad Control	157
9.1.6	Identification	158
9.2	Initialization and Configuration	158
9.3	Register Map	159
9.4	Register Descriptions	161
10	Timers	196
10.1	Block Diagram	197

10.2	Functional Description	197
10.2.1	GPTM Reset Conditions	197
10.2.2	32-Bit Timer Operating Modes	197
10.2.3	16-Bit Timer Operating Modes	199
10.3	Initialization and Configuration	203
10.3.1	32-Bit One-Shot/Periodic Timer Mode	203
10.3.2	32-Bit Real-Time Clock (RTC) Mode	204
10.3.3	16-Bit One-Shot/Periodic Timer Mode	204
10.3.4	16-Bit Input Edge Count Mode	205
10.3.5	16-Bit Input Edge Timing Mode	205
10.3.6	16-Bit PWM Mode	206
10.4	Register Map	206
10.5	Register Descriptions	207
11	Watchdog Timer	229
11.1	Block Diagram	229
11.2	Functional Description	229
11.3	Initialization and Configuration	230
11.4	Register Map	230
11.5	Register Descriptions	231
12	ADC	252
12.1	Block Diagram	253
12.2	Functional Description	253
12.2.1	Sample Sequencers	253
12.2.2	Module Control	254
12.2.3	Hardware Sample Averaging Circuit	255
12.2.4	Analog-to-Digital Converter	255
12.2.5	Test Modes	255
12.2.6	Internal Temperature Sensor	255
12.3	Initialization and Configuration	256
12.3.1	Module Initialization	256
12.3.2	Sample Sequencer Configuration	256
12.4	Register Map	257
12.5	Register Descriptions	258
13	UART	285
13.1	Block Diagram	286
13.2	Functional Description	286
13.2.1	Transmit/Receive Logic	286
13.2.2	Baud-Rate Generation	287
13.2.3	Data Transmission	288
13.2.4	Serial IR (SIR)	288
13.2.5	FIFO Operation	289
13.2.6	Interrupts	289
13.2.7	Loopback Operation	290
13.2.8	IrDA SIR block	290
13.3	Initialization and Configuration	290
13.4	Register Map	291
13.5	Register Descriptions	292

14	SSI	325
14.1	Block Diagram	325
14.2	Functional Description	326
14.2.1	Bit Rate Generation	326
14.2.2	FIFO Operation	326
14.2.3	Interrupts	326
14.2.4	Frame Formats	327
14.3	Initialization and Configuration	334
14.4	Register Map	335
14.5	Register Descriptions	336
15	Inter-Integrated Circuit (I²C) Interface	359
15.1	Block Diagram	359
15.2	Functional Description	359
15.2.1	I ² C Bus Functional Overview	360
15.2.2	Available Speed Modes	362
15.2.3	Interrupts	363
15.2.4	Loopback Operation	363
15.2.5	Command Sequence Flow Charts	364
15.3	Initialization and Configuration	370
15.4	I ² C Register Map	371
15.5	I ² C Master	372
15.6	I ² C Slave	385
16	CAN	394
16.1	Controller Area Network Overview	394
16.2	Controller Area Network Features	394
16.3	Controller Area Network Block Diagram	395
16.4	Controller Area Network Functional Description	396
16.4.1	Initialization	396
16.4.2	Operation	397
16.4.3	Transmitting Message Objects	397
16.4.4	Configuring a Transmit Message Object	397
16.4.5	Updating a Transmit Message Object	398
16.4.6	Accepting Received Message Objects	398
16.4.7	Receiving a Data Frame	399
16.4.8	Receiving a Remote Frame	399
16.4.9	Receive/Transmit Priority	399
16.4.10	Configuring a Receive Message Object	399
16.4.11	Handling of Received Message Objects	400
16.4.12	Handling of Interrupts	400
16.4.13	Bit Timing Configuration Error Considerations	401
16.4.14	Bit Time and Bit Rate	401
16.4.15	Calculating the Bit Timing Parameters	403
16.5	Controller Area Network Register Map	405
16.6	Register Descriptions	407
17	Ethernet	438
17.1	Block Diagram	439
17.2	Functional Description	439

17.2.1	Internal MII Operation	439
17.2.2	PHY Configuration/Operation	440
17.2.3	MAC Configuration/Operation	441
17.2.4	Interrupts	443
17.3	Initialization and Configuration	444
17.4	Ethernet Register Map	444
17.5	Ethernet MAC	446
17.6	MIU Management	463
18	Analog Comparators	482
18.1	Block Diagram	483
18.2	Functional Description	483
18.2.1	Internal Reference Programming	485
18.3	Initialization and Configuration	486
18.4	Register Map	486
18.5	Register Descriptions	487
19	Pin Diagram	495
20	Signal Tables	496
21	Operating Characteristics	510
22	Electrical Characteristics	511
22.1	DC Characteristics	511
22.1.1	Maximum Ratings	511
22.1.2	Recommended DC Operating Conditions	511
22.1.3	On-Chip Low Drop-Out (LDO) Regulator Characteristics	512
22.1.4	Power Specifications	512
22.1.5	Flash Memory Characteristics	513
22.2	AC Characteristics	513
22.2.1	Load Conditions	513
22.2.2	Clocks	513
22.2.3	Temperature Sensor	514
22.2.4	Analog-to-Digital Converter	514
22.2.5	Analog Comparator	515
22.2.6	I ² C	515
22.2.7	Ethernet Controller	516
22.2.8	Hibernation Module	519
22.2.9	Synchronous Serial Interface (SSI)	519
22.2.10	JTAG and Boundary Scan	521
22.2.11	General-Purpose I/O	522
22.2.12	Reset	523
23	Package Information	525
24	Ordering Information	527
24.1	Ordering Information	527
24.2	Company Information	527
24.3	Support Information	527
A	Serial Flash Loader	528
A.1	Serial Flash Loader	528
A.2	Interfaces	528

A.2.1	UART	528
A.2.2	SSI	528
A.3	Packet Handling	529
A.3.1	Packet Format	529
A.3.2	Sending Packets	529
A.3.3	Receiving Packets	529
A.4	Commands	530
A.4.1	COMMAND_PING (0X20)	530
A.4.2	COMMAND_GET_STATUS (0x23)	530
A.4.3	COMMAND_DOWNLOAD (0x21)	530
A.4.4	COMMAND_SEND_DATA (0x24)	531
A.4.5	COMMAND_RUN (0x22)	531
A.4.6	COMMAND_RESET (0x25)	531
B	Register Quick Reference	533

DRAFT

List of Figures

Figure 1-1.	Stellaris® Fury-class High-Level Block Diagram	28
Figure 2-1.	CPU Block Diagram	36
Figure 2-2.	TPIU Block Diagram	37
Figure 5-1.	JTAG Module Block Diagram	47
Figure 5-2.	Test Access Port State Machine	50
Figure 5-3.	IDCODE Register Format	55
Figure 5-4.	BYPASS Register Format	56
Figure 5-5.	Boundary Scan Register Format	56
Figure 6-1.	External Circuitry to Extend Reset	58
Figure 7-1.	Hibernation Module Block Diagram	114
Figure 8-1.	Flash Block Diagram	131
Figure 9-1.	GPIO DATA Write Example	156
Figure 9-2.	GPIO DATA Read Example	156
Figure 10-1.	GPTM Module Block Diagram	197
Figure 10-2.	16-Bit Input Edge Count Mode Example	201
Figure 10-3.	16-Bit Input Edge Time Mode Example	202
Figure 10-4.	16-Bit PWM Mode Example	203
Figure 11-1.	WDT Module Block Diagram	229
Figure 12-1.	ADC Module Block Diagram	253
Figure 12-2.	Internal Temperature Sensor Characteristic	256
Figure 13-1.	UART Module Block Diagram	286
Figure 13-2.	UART Character Frame	287
Figure 13-3.	IrDA Data Modulation	289
Figure 14-1.	SSI Module Block Diagram	325
Figure 14-2.	TI Synchronous Serial Frame Format (Single Transfer)	328
Figure 14-3.	TI Synchronous Serial Frame Format (Continuous Transfer)	328
Figure 14-4.	Freescall SPI Format (Single Transfer) with SPO=0 and SPH=0	329
Figure 14-5.	Freescall SPI Format (Continuous Transfer) with SPO=0 and SPH=0	329
Figure 14-6.	Freescall SPI Frame Format with SPO=0 and SPH=1	330
Figure 14-7.	Freescall SPI Frame Format (Single Transfer) with SPO=1 and SPH=0	331
Figure 14-8.	Freescall SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0	331
Figure 14-9.	Freescall SPI Frame Format with SPO=1 and SPH=1	332
Figure 14-10.	MICROWIRE Frame Format (Single Frame)	333
Figure 14-11.	MICROWIRE Frame Format (Continuous Transfer)	334
Figure 14-12.	MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements	334
Figure 15-1.	I ² C Block Diagram	359
Figure 15-2.	I ² C Bus Configuration	360
Figure 15-3.	START and STOP Conditions	360
Figure 15-4.	Complete Data Transfer with a 7-Bit Address	361
Figure 15-5.	R/S Bit in First Byte	361
Figure 15-6.	Data Validity During Bit Transfer on the I ² C Bus	361
Figure 15-7.	Master Single SEND	364
Figure 15-8.	Master Single RECEIVE	365
Figure 15-9.	Master Burst SEND	366
Figure 15-10.	Master Burst RECEIVE	367
Figure 15-11.	Master Burst RECEIVE after Burst SEND	368

Figure 15-12. Master Burst SEND after Burst RECEIVE	369
Figure 15-13. Slave Command Sequence	370
Figure 16-1. CAN Module Block Diagram	395
Figure 16-2. CAN Bit Time	402
Figure 17-1. Ethernet Controller Block Diagram	439
Figure 17-2. Ethernet Controller	439
Figure 17-3. Ethernet Frame	441
Figure 18-1. Analog Comparator Module Block Diagram	483
Figure 18-2. Structure of Comparator Unit	484
Figure 18-3. Comparator Internal Reference Structure	485
Figure 19-1. Pin Connection Diagram	495
Figure 22-1. Load Conditions	513
Figure 22-2. I ² C Timing	516
Figure 22-3. External XTLP Oscillator Characteristics	518
Figure 22-4. Hibernation Module Timing	519
Figure 22-5. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement	520
Figure 22-6. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer	520
Figure 22-7. SSI Timing for SPI Frame Format (FRF=00), with SPH=1	521
Figure 22-8. JTAG Test Clock Input Timing	522
Figure 22-9. JTAG Test Access Port (TAP) Timing	522
Figure 22-10. JTAG TRST Timing	522
Figure 22-11. External Reset Timing ($\overline{\text{RST}}$)	523
Figure 22-12. Power-On Reset Timing	524
Figure 22-13. Brown-Out Reset Timing	524
Figure 22-14. Software Reset Timing	524
Figure 22-15. Watchdog Reset Timing	524
Figure 23-1. 100-Pin LQFP Package	525

List of Tables

Table 1.	Documentation Conventions	20
Table 3-1.	Memory Map	41
Table 4-1.	Exception Types	43
Table 4-2.	Interrupts	44
Table 5-1.	JTAG Port Pins Reset State	48
Table 5-2.	JTAG Instruction Register Commands	53
Table 6-1.	System Control Register Map	63
Table 6-2.	VADJ to VOUT	68
Table 6-3.	Default Crystal Field Values and PLL Programming	75
Table 7-1.	Hibernation Module Register Map	118
Table 8-1.	Flash Protection Policy Combinations	133
Table 8-2.	Flash Resident Registers	134
Table 8-3.	Internal Memory Register Map	134
Table 9-1.	GPIO Pad Configuration Examples	158
Table 9-2.	GPIO Interrupt Configuration Example	158
Table 9-3.	GPIO Register Map	160
Table 10-1.	16-Bit Timer With Prescaler Configurations	200
Table 10-2.	Timers Register Map	206
Table 11-1.	Watchdog Timer Register Map	230
Table 12-1.	Samples and FIFO Depth of Sequencers	253
Table 12-2.	ADC Register Map	257
Table 13-1.	UART Register Map	291
Table 14-1.	SSI Register Map	335
Table 15-1.	Examples of I ² C Master Timer Period versus Speed Mode	362
Table 15-2.	Inter-Integrated Circuit (I ² C) Interface Register Map	371
Table 15-3.	Write Field Decoding for I2CMCS[3:0] Field (Sheet 1 of 3)	376
Table 16-1.	Transmit Message Object Bit Settings	398
Table 16-2.	Receive Message Object Bit Settings	400
Table 16-3.	CAN Protocol Ranges	402
Table 16-4.	CAN Register Map	405
Table 17-1.	TX & RX FIFO Organization	442
Table 17-2.	Ethernet Register Map	445
Table 18-1.	Comparator 0 Operating Modes	484
Table 18-2.	Comparator 1 Operating Modes	484
Table 18-3.	Comparator 2 Operating Modes	485
Table 18-4.	Internal Reference Voltage and ACREFACTL Field Values	485
Table 18-5.	Analog Comparators Register Map	487
Table 20-1.	Signals by Pin Number	496
Table 20-2.	Signals by Signal Name	500
Table 20-3.	Signals by Function, Except for GPIO	504
Table 20-4.	GPIO Pins and Alternate Functions	508
Table 21-1.	Temperature Characteristics	510
Table 21-2.	Thermal Characteristics	510
Table 22-1.	Maximum Ratings	511
Table 22-2.	Recommended DC Operating Conditions	511
Table 22-3.	LDO Regulator Characteristics	512

Table 22-4.	Flash Memory Characteristics	513
Table 22-5.	Phase Locked Loop (PLL) Characteristics	513
Table 22-6.	Clock Characteristics	513
Table 22-7.	Crystal Characteristics	514
Table 22-8.	Temperature Sensor Characteristics	514
Table 22-9.	ADC Characteristics	514
Table 22-10.	Analog Comparator Characteristics	515
Table 22-11.	Analog Comparator Voltage Reference Characteristics	515
Table 22-12.	I ² C Characteristics	515
Table 22-13.	100BASE-TX Transmitter Characteristics	516
Table 22-14.	100BASE-TX Transmitter Characteristics (informative)	516
Table 22-15.	100BASE-TX Receiver Characteristics	516
Table 22-16.	10BASE-T Transmitter Characteristics	516
Table 22-17.	10BASE-T Transmitter Characteristics (informative)	517
Table 22-18.	10BASE-T Receiver Characteristics	517
Table 22-19.	Isolation Transformers	517
Table 22-20.	Ethernet Reference Crystal	518
Table 22-21.	External XTLP Oscillator Characteristics	518
Table 22-22.	Hibernation Module Characteristics	519
Table 22-23.	SSI Characteristics	519
Table 22-24.	JTAG Characteristics	521
Table 22-25.	GPIO Characteristics	523
Table 22-26.	Reset Characteristics	523
Table 24-1.	Part Ordering Information	527

List of Registers

System Control	57
Register 1: Device Identification 0 (DID0), offset 0x000	65
Register 2: Brown-Out Reset Control (PBORCTL), offset 0x030	67
Register 3: LDO Power Control (LDOPCTL), offset 0x034	68
Register 4: Raw Interrupt Status (RIS), offset 0x050	69
Register 5: Interrupt Mask Control (IMC), offset 0x054	70
Register 6: Masked Interrupt Status and Clear (MISC), offset 0x058	71
Register 7: Reset Cause (RESC), offset 0x05C	72
Register 8: Run-Mode Clock Configuration (RCC), offset 0x060	73
Register 9: XTAL to PLL Translation (PLLCFG), offset 0x064	77
Register 10: Run-Mode Clock Configuration 2 (RCC2), offset 0x070	78
Register 11: Deep Sleep Clock Configuration (DSLPCCLKCFG), offset 0x144	80
Register 12: Device Identification 1 (DID1), offset 0x004	81
Register 13: Device Capabilities 0 (DC0), offset 0x008	83
Register 14: Device Capabilities 1 (DC1), offset 0x010	84
Register 15: Device Capabilities 2 (DC2), offset 0x014	86
Register 16: Device Capabilities 3 (DC3), offset 0x018	88
Register 17: Device Capabilities 4 (DC4), offset 0x01C	90
Register 18: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100	91
Register 19: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110	93
Register 20: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120	95
Register 21: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104	97
Register 22: Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114	99
Register 23: Deep Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124	101
Register 24: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108	103
Register 25: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118	105
Register 26: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128	107
Register 27: Software Reset Control 0 (SRCR0), offset 0x040	109
Register 28: Software Reset Control 1 (SRCR1), offset 0x044	110
Register 29: Software Reset Control 2 (SRCR2), offset 0x048	112
Hibernation Module	113
Register 1: Hibernation RTC Counter (HIBRTCC), offset 0x000	119
Register 2: Hibernation RTC Match 0 (HIBRTCM0), offset 0x004	120
Register 3: Hibernation RTC Match 1 (HIBRTCM1), offset 0x008	121
Register 4: Hibernation RTC Load (HIBRTCLD), offset 0x00C	122
Register 5: Hibernation Control (HIBCTL), offset 0x010	123
Register 6: Hibernation Interrupt Mask (HIBIM), offset 0x014	125
Register 7: Hibernation Raw Interrupt Status (HIBRIS), offset 0x018	126
Register 8: Hibernation Masked Interrupt Status (HIBMIS), offset 0x01C	127
Register 9: Hibernation Interrupt Clear (HIBIC), offset 0x020	128
Register 10: Hibernation RTC Trim (HIBRTCT), offset 0x024	129
Register 11: Hibernation Data (HIBDATA), offset 0x030-0x12C	130
Internal Memory	131
Register 1: Flash Memory Address (FMA), offset 0x000	136
Register 2: Flash Memory Data (FMD), offset 0x004	137

Register 3:	Flash Memory Control (FMC), offset 0x008	138
Register 4:	Flash Controller Raw Interrupt Status (FCRIS), offset 0x00C	140
Register 5:	Flash Controller Interrupt Mask (FCIM), offset 0x010	141
Register 6:	Flash Controller Masked Interrupt Status and Clear (FCMISC), offset 0x014	142
Register 7:	USec Reload (USECRL), offset 0x140	143
Register 8:	Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200	144
Register 9:	Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400	145
Register 10:	User Debug (USER_DBG), offset 0x1D0	146
Register 11:	User Register 0 (USER_REG0), offset 0x1E0	147
Register 12:	User Register 1 (USER_REG1), offset 0x1E4	148
Register 13:	Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204	149
Register 14:	Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208	150
Register 15:	Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C	151
Register 16:	Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404	152
Register 17:	Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408	153
Register 18:	Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C	154
GPIO		155
Register 1:	GPIO Data (GPIODATA), offset 0x000	162
Register 2:	GPIO Direction (GPIODIR), offset 0x400	163
Register 3:	GPIO Interrupt Sense (GPIOIS), offset 0x404	164
Register 4:	GPIO Interrupt Both Edges (GPIOIBE), offset 0x408	165
Register 5:	GPIO Interrupt Event (GPIOIEV), offset 0x40C	166
Register 6:	GPIO Interrupt Mask (GPIOIM), offset 0x410	167
Register 7:	GPIO Raw Interrupt Status (GPIORIS), offset 0x414	168
Register 8:	GPIO Masked Interrupt Status (GPIOMIS), offset 0x418	169
Register 9:	GPIO Interrupt Clear (GPIOICR), offset 0x41C	170
Register 10:	GPIO Alternate Function Select (GPIOAFSEL), offset 0x420	171
Register 11:	GPIO 2-mA Drive Select (GPIODR2R), offset 0x500	173
Register 12:	GPIO 4-mA Drive Select (GPIODR4R), offset 0x504	174
Register 13:	GPIO 8-mA Drive Select (GPIODR8R), offset 0x508	175
Register 14:	GPIO Open Drain Select (GPIOODR), offset 0x50C	176
Register 15:	GPIO Pull-Up Select (GPIOPUR), offset 0x510	177
Register 16:	GPIO Pull-Down Select (GPIOPDR), offset 0x514	178
Register 17:	GPIO Slew Rate Control Select (GPIOSLR), offset 0x518	179
Register 18:	GPIO Digital Enable (GPIODEN), offset 0x51C	180
Register 19:	GPIO Lock (GPIOLOCK), offset 0x520	181
Register 20:	GPIO Commit (GPIOCR), offset 0x524	182
Register 21:	GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0	184
Register 22:	GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4	185
Register 23:	GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8	186
Register 24:	GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC	187
Register 25:	GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0	188
Register 26:	GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4	189
Register 27:	GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8	190
Register 28:	GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC	191
Register 29:	GPIO PrimeCell Identification 0 (GPIOPCellID0), offset 0xFF0	192
Register 30:	GPIO PrimeCell Identification 1 (GPIOPCellID1), offset 0xFF4	193
Register 31:	GPIO PrimeCell Identification 2 (GPIOPCellID2), offset 0xFF8	194

Register 32: GPIO PrimeCell Identification 3 (GPIOCellID3), offset 0xFFC	195
Timers	196
Register 1: GPTM Configuration (GPTMCFG), offset 0x000	208
Register 2: GPTM TimerA Mode (GPTMTAMR), offset 0x004	209
Register 3: GPTM TimerB Mode (GPTMTBMR), offset 0x008	210
Register 4: GPTM Control (GPTMCTL), offset 0x00C	211
Register 5: GPTM Interrupt Mask (GPTMIMR), offset 0x018	213
Register 6: GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C	215
Register 7: GPTM Masked Interrupt Status (GPTMMIS), offset 0x020	216
Register 8: GPTM Interrupt Clear (GPTMICR), offset 0x024	217
Register 9: GPTM TimerA Interval Load (GPTMTAILR), offset 0x028	219
Register 10: GPTM TimerB Interval Load (GPTMTBILR), offset 0x02C	220
Register 11: GPTM TimerA Match (GPTMTAMATCHR), offset 0x030	221
Register 12: GPTM TimerB Match (GPTMTBMATCHR), offset 0x034	222
Register 13: GPTM TimerA Prescale (GPTMTAPR), offset 0x038	223
Register 14: GPTM TimerB Prescale (GPTMTBPR), offset 0x03C	224
Register 15: GPTM TimerA Prescale Match (GPTMTAPMR), offset 0x040	225
Register 16: GPTM TimerB Prescale Match (GPTMTBPMR), offset 0x044	226
Register 17: GPTM TimerA (GPTMTAR), offset 0x048	227
Register 18: GPTM TimerB (GPTMTBR), offset 0x04C	228
Watchdog Timer	229
Register 1: Watchdog Load (WDTLOAD), offset 0x000	232
Register 2: Watchdog Value (WDTVALUE), offset 0x004	233
Register 3: Watchdog Control (WDTCTL), offset 0x008	234
Register 4: Watchdog Interrupt Clear (WDTICR), offset 0x00C	235
Register 5: Watchdog Raw Interrupt Status (WDTRIS), offset 0x010	236
Register 6: Watchdog Masked Interrupt Status (WDTMIS), offset 0x014	237
Register 7: Watchdog Test (WDTTEST), offset 0x418	238
Register 8: Watchdog Lock (WDTLOCK), offset 0xC00	239
Register 9: Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0	240
Register 10: Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4	241
Register 11: Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8	242
Register 12: Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC	243
Register 13: Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0	244
Register 14: Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4	245
Register 15: Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8	246
Register 16: Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC	247
Register 17: Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0	248
Register 18: Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4	249
Register 19: Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8	250
Register 20: Watchdog PrimeCell Identification 3 (WDTPCellID3), offset 0xFFC	251
ADC	252
Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000	259
Register 2: ADC Raw Interrupt Status (ADCRIS), offset 0x004	260
Register 3: ADC Interrupt Mask (ADCIM), offset 0x008	261
Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C	262
Register 5: ADC Overflow Status (ADCOSTAT), offset 0x010	263
Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014	264

Register 7:	ADC Underflow Status (ADCUSTAT), offset 0x018	265
Register 8:	ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020	266
Register 9:	ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028	267
Register 10:	ADC Sample Averaging Control (ADCSAC), offset 0x030	268
Register 11:	ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040	269
Register 12:	ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044	271
Register 13:	ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048	273
Register 14:	ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068	273
Register 15:	ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088	273
Register 16:	ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C	274
Register 17:	ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C	274
Register 18:	ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C	274
Register 19:	ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060	275
Register 20:	ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064	276
Register 21:	ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080	277
Register 22:	ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084	278
Register 23:	ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0	279
Register 24:	ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4	280
Register 25:	ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8	281
Register 26:	ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC	282
Register 27:	ADC Test Mode Loopback (ADCTMLB), offset 0x100	283
UART		285
Register 1:	UART Data (UARTDR), offset 0x000	293
Register 2:	UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004	295
Register 3:	UART Flag (UARTFR), offset 0x018	297
Register 4:	UART IrDA Low-Power Register (UARTILPR), offset 0x020	299
Register 5:	UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024	300
Register 6:	UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028	301
Register 7:	UART Line Control (UARTLCRH), offset 0x02C	302
Register 8:	UART Control (UARTCTL), offset 0x030	304
Register 9:	UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034	306
Register 10:	UART Interrupt Mask (UARTIM), offset 0x038	307
Register 11:	UART Raw Interrupt Status (UARTRIS), offset 0x03C	309
Register 12:	UART Masked Interrupt Status (UARTMIS), offset 0x040	310
Register 13:	UART Interrupt Clear (UARTICR), offset 0x044	311
Register 14:	UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0	313
Register 15:	UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4	314
Register 16:	UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8	315
Register 17:	UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC	316
Register 18:	UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0	317
Register 19:	UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4	318
Register 20:	UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8	319
Register 21:	UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC	320
Register 22:	UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0	321
Register 23:	UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4	322
Register 24:	UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8	323
Register 25:	UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC	324

SSI	325
Register 1: SSI Control 0 (SSICR0), offset 0x000	337
Register 2: SSI Control 1 (SSICR1), offset 0x004	339
Register 3: SSI Data (SSIDR), offset 0x008	340
Register 4: SSI Status (SSISR), offset 0x00C	341
Register 5: SSI Clock Prescale (SSICPSR), offset 0x010	342
Register 6: SSI Interrupt Mask (SSIIM), offset 0x014	343
Register 7: SSI Raw Interrupt Status (SSIRIS), offset 0x018	344
Register 8: SSI Masked Interrupt Status (SSIMIS), offset 0x01C	345
Register 9: SSI Interrupt Clear (SSIICR), offset 0x020	346
Register 10: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0	347
Register 11: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4	348
Register 12: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8	349
Register 13: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC	350
Register 14: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0	351
Register 15: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4	352
Register 16: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8	353
Register 17: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC	354
Register 18: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0	355
Register 19: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4	356
Register 20: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8	357
Register 21: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC	358
Inter-Integrated Circuit (I²C) Interface	359
Register 1: I ² C Master Slave Address (I2CMSA), offset 0x000	373
Register 2: I ² C Master Control/Status (I2CMCS), offset 0x004	374
Register 3: I ² C Master Data (I2CMDR), offset 0x008	378
Register 4: I ² C Master Timer Period (I2CMTPR), offset 0x00C	379
Register 5: I ² C Master Interrupt Mask (I2CMIMR), offset 0x010	380
Register 6: I ² C Master Raw Interrupt Status (I2CMRIS), offset 0x014	381
Register 7: I ² C Master Masked Interrupt Status (I2CMMIS), offset 0x018	382
Register 8: I ² C Master Interrupt Clear (I2CMICR), offset 0x01C	383
Register 9: I ² C Master Configuration (I2CMCR), offset 0x020	384
Register 10: I ² C Slave Own Address (I2CSOAR), offset 0x000	386
Register 11: I ² C Slave Control/Status (I2CSCSR), offset 0x004	387
Register 12: I ² C Slave Data (I2CSDR), offset 0x008	389
Register 13: I ² C Slave Interrupt Mask (I2CSIMR), offset 0x00C	390
Register 14: I ² C Slave Raw Interrupt Status (I2CSRIS), offset 0x010	391
Register 15: I ² C Slave Masked Interrupt Status (I2CSMIS), offset 0x014	392
Register 16: I ² C Slave Interrupt Clear (I2CSICR), offset 0x018	393
CAN	394
Register 1: CAN Control (CANCTL), offset 0x000	408
Register 2: CAN Status (CANSTS), offset 0x004	410
Register 3: CAN Error Counter (CANERR), offset 0x008	413
Register 4: CAN Bit Timing (CANBIT), offset 0x00C	414
Register 5: CAN Interrupt (CANINT), offset 0x010	416
Register 6: CAN Test (CANTST), offset 0x014	417
Register 7: CAN Baud Rate Prescaler Extension (CANBRPE), offset 0x018	419

Register 8:	CAN IF1 Command Request (CANIF1CRQ), offset 0x020	420
Register 9:	CAN IF2 Command Request (CANIF2CRQ), offset 0x080	420
Register 10:	CAN IF1 Command Mask (CANIF1CMSK), offset 0x024	421
Register 11:	CAN IF2 Command Mask (CANIF2CMSK), offset 0x084	421
Register 12:	CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028	424
Register 13:	CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088	424
Register 14:	CAN IF1 Mask 2 (CANIF1MSK2), offset 0x02C	425
Register 15:	CAN IF2 Mask 2 (CANIF2MSK2), offset 0x08C	425
Register 16:	CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030	426
Register 17:	CAN IF2 Arbitration 1 (CANIF2ARB1), offset 0x090	426
Register 18:	CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034	427
Register 19:	CAN IF2 Arbitration 2 (CANIF2ARB2), offset 0x094	427
Register 20:	CAN IF1 Message Control (CANIF1MCTL), offset 0x038	428
Register 21:	CAN IF2 Message Control (CANIF2MCTL), offset 0x098	428
Register 22:	CAN IF1 Data A1 (CANIF1DA1), offset 0x03C	430
Register 23:	CAN IF2 Data A1 (CANIF2DA1), offset 0x09C	430
Register 24:	CAN IF1 Data A2 (CANIF1DA2), offset 0x040	431
Register 25:	CAN IF2 Data A2 (CANIF2DA2), offset 0x0A0	431
Register 26:	CAN IF1 Data B1 (CANIF1DB1), offset 0x044	432
Register 27:	CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4	432
Register 28:	CAN IF1 Data B2 (CANIF1DB2), offset 0x048	433
Register 29:	CAN IF2 Data B2 (CANIF2DB2), offset 0x0A8	433
Register 30:	CAN Transmission Request 1 (CANTXRQ1), offset 0x100	434
Register 31:	CAN Transmission Request 2 (CANTXRQ2), offset 0x104	434
Register 32:	CAN New Data 1 (CANNWDA1), offset 0x120	435
Register 33:	CAN New Data 2 (CANNWDA2), offset 0x124	435
Register 34:	CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140	436
Register 35:	CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144	436
Register 36:	CAN Message 1 Valid (CANMSG1VAL), offset 0x160	437
Register 37:	CAN Message 2 Valid (CANMSG2VAL), offset 0x164	437
Ethernet		438
Register 1:	Ethernet MAC Raw Interrupt Status (MACRIS), offset 0x000	447
Register 2:	Ethernet MAC Interrupt Acknowledge (MACIACK), offset 0x000	449
Register 3:	Ethernet MAC Interrupt Mask (MACIM), offset 0x004	450
Register 4:	Ethernet MAC Receive Control (MACRCTL), offset 0x008	451
Register 5:	Ethernet MAC Transmit Control (MACTCTL), offset 0x00C	452
Register 6:	Ethernet MAC Data (MACDATA), offset 0x010	453
Register 7:	Ethernet MAC Individual Address 0 (MACIA0), offset 0x014	454
Register 8:	Ethernet MAC Individual Address 1 (MACIA1), offset 0x018	455
Register 9:	Ethernet MAC Threshold (MACTHR), offset 0x01C	456
Register 10:	Ethernet MAC Management Control (MACMCTL), offset 0x020	457
Register 11:	Ethernet MAC Management Divider (MACMDV), offset 0x024	458
Register 12:	Ethernet MAC Management Address (MACMADD), offset 0x028	459
Register 13:	Ethernet MAC Management Transmit Data (MACMTXD), offset 0x02C	460
Register 14:	Ethernet MAC Management Receive Data (MACMRXD), offset 0x030	461
Register 15:	Ethernet MAC Number of Packets (MACNP), offset 0x034	462
Register 16:	Ethernet MAC Transmission Request (MACTR), offset 0x038	463
Register 17:	Ethernet PHY Management Register 0 – Control (MR0), offset 0x00	464

Register 18:	Ethernet PHY Management Register 1 – Status (MR1), offset 0x01	466
Register 19:	Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2), offset 0x02	468
Register 20:	Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3), offset 0x03	469
Register 21:	Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4), offset 0x04	470
Register 22:	Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5), offset 0x05	472
Register 23:	Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6), offset 0x06	473
Register 24:	Ethernet PHY Management Register 16 – Vendor-Specific (MR16), offset 0x10	474
Register 25:	Ethernet PHY Management Register 17 – Interrupt Control/Status (MR17), offset 0x11	476
Register 26:	Ethernet PHY Management Register 18 – Diagnostic (MR18), offset 0x12	478
Register 27:	Ethernet PHY Management Register 19 – Transceiver Control (MR19), offset 0x13	479
Register 28:	Ethernet PHY Management Register 23 – LED Configuration (MR23), offset 0x17	480
Register 29:	Ethernet PHY Management Register 24 –MDI/MDIX Control (MR24), offset 0x18	481
Analog Comparators		482
Register 1:	Analog Comparator Masked Interrupt Status (ACMIS), offset 0x00	488
Register 2:	Analog Comparator Raw Interrupt Status (ACRIS), offset 0x04	489
Register 3:	Analog Comparator Interrupt Enable (ACINTEN), offset 0x08	490
Register 4:	Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x10	491
Register 5:	Analog Comparator Status 0 (ACSTAT0), offset 0x20	492
Register 6:	Analog Comparator Status 1 (ACSTAT1), offset 0x40	492
Register 7:	Analog Comparator Status 2 (ACSTAT2), offset 0x60	492
Register 8:	Analog Comparator Control 0 (ACCTL0), offset 0x24	493
Register 9:	Analog Comparator Control 1 (ACCTL1), offset 0x44	493
Register 10:	Analog Comparator Control 2 (ACCTL2), offset 0x64	493

About This Document

This data sheet provides reference information for the LM3S8938 microcontroller, describing the functional blocks of the system-on-chip (SoC) device designed around the ARM® Cortex™-M3 core.

Audience

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

About This Manual

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

Related Documents

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

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

The following related documents are also referenced:

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

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

Documentation Conventions

This document uses the conventions shown in Table 1 on page 20.

Table 1. 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 0xnnn	A hexadecimal increment to a register's address, relative to that module's base address as specified in "Memory Map" on page 41.
Register N	Registers are numbered consecutively throughout the document to aid in referencing them. The register number has no meaning to software.

Notation	Meaning
reserved	Register bits marked <i>reserved</i> are reserved for future use. In most cases, reserved bits are set to 0; however, user software should not rely on the value of a reserved bit. To provide software compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
yy:xx	The range of register bits inclusive from xx to yy. For example, 31:15 means bits 15 through 31 in that register.
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/W1C	Software can read or write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged. This register type is primarily used for clearing interrupt status bits where the read operation provides the interrupt status and the write of the read value clears only the interrupts being reported at the time the register was read.
W1C	Software can write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged. A read of the register returns no meaningful data. This register is typically used to clear the corresponding bit in an interrupt register.
WO	Only a write by software is valid; a read of the register returns no meaningful data.
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.
assert a signal	Change the value of the signal from the logically False state to the logically True state. For active High signals, the asserted signal value is 1 (High); for active Low signals, the asserted signal value is 0 (Low). The active polarity (High or Low) is defined by the signal name (see <code>SIGNAL</code> and <code>SIGNAL</code> below).
deassert a signal	Change the value of the signal from the logically True state to the logically False state.
<code>SIGNAL</code>	Signal names are in uppercase and in the Courier font. An overbar on a signal name indicates that it is active Low. To assert <code>SIGNAL</code> is to drive it Low; to deassert <code>SIGNAL</code> is to drive it High.
<code>SIGNAL</code>	Signal names are in uppercase and in the Courier font. An active High signal has no overbar. To assert <code>SIGNAL</code> is to drive it High; to deassert <code>SIGNAL</code> is to drive it Low.
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, 0x00FF is the hexadecimal number FF. Binary numbers are indicated with a b suffix, for example, 1011b. Decimal numbers are written without a prefix or suffix.

1 Architectural Overview

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

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

The LM3S8938 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 LM3S8938 microcontroller features a Battery-backed Hibernation module to efficiently power down the LM3S8938 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 LM3S8938 microcontroller perfectly for battery applications.

In addition, the LM3S8938 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 LM3S8938 microcontroller is code-compatible to all members of the extensive Stellaris[®] family; providing flexibility to fit our customers' precise needs.

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

1.1 Product Features

The LM3S8938 microcontroller includes the following product features:

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

- Hardware-division and single-cycle-multiplication
- Integrated Nested Vectored Interrupt Controller (NVIC) providing deterministic interrupt handling
- 34 interrupts with eight priority levels
- Memory protection unit (MPU), providing a privileged mode for protected operating system functionality
- Unaligned data access, enabling data to be efficiently packed into memory
- Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
- Internal Memory
 - 256 KB single-cycle flash
 - User-managed flash block protection on a 2-KB block basis
 - User-managed flash data programming
 - User-defined and managed flash-protection block
 - 64 KB single-cycle SRAM
- General-Purpose Timers
 - Four General-Purpose Timer Modules (GPTM), each of which provides two 16-bit timer/counters. Each GPTM can be configured to operate independently as timers or event counters (eight total): as a single 32-bit timer (four total), as one 32-bit Real-Time Clock (RTC) to event capture, for Pulse Width Modulation (PWM), or to trigger analog-to-digital conversions
 - 32-bit Timer modes
 - Programmable one-shot timer
 - Programmable periodic timer
 - Real-Time Clock when using an external 32.768-KHz clock as the input
 - User-enabled stalling in periodic and one-shot mode when the controller asserts the CPU Halt flag during debug
 - ADC event trigger
 - 16-bit Timer modes
 - General-purpose timer function with an 8-bit prescaler
 - Programmable one-shot timer
 - Programmable periodic timer
 - User-enabled stalling when the controller asserts CPU Halt flag during debug

- ADC event trigger
- 16-bit Input Capture modes
 - Input edge count capture
 - Input edge time capture
- 16-bit PWM mode
 - Simple PWM mode with software-programmable output inversion of the PWM signal
- ARM FiRM-compliant Watchdog Timer
 - 32-bit down counter with a programmable load register
 - Separate watchdog clock with an enable
 - Programmable interrupt generation logic with interrupt masking
 - Lock register protection from runaway software
 - Reset generation logic with an enable/disable
 - User-enabled stalling when the controller asserts the CPU Halt flag during debug
- Controller Area Network (CAN)
 - Supports CAN protocol version 2.0 part A/B
 - Bit rates up to 1Mb/s
 - 32 message objects, each with its own identifier mask
 - Maskable interrupt
 - Disable automatic retransmission mode for TTCAN
 - Programmable loop-back mode for self-test operation
- 10/100 Ethernet Controller
 - Conforms to the IEEE 802.3-2002 Specification
 - Full- and half-duplex for both 100 Mbps and 10 Mbps operation
 - Integrated 10/100 Mbps Transceiver (PHY)
 - Automatic MDI/MDI-X cross-over correction
 - Programmable MAC address
 - Power-saving and power-down modes
- Synchronous Serial Interface (SSI)
 - Master or slave operation

- Programmable clock bit rate and prescale
- Separate transmit and receive FIFOs, 16 bits wide, 8 locations deep
- Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces
- Programmable data frame size from 4 to 16 bits
- Internal loopback test mode for diagnostic/debug testing
- UART
 - Three fully programmable 16C550-type UARTs with IrDA support
 - Separate 16x8 transmit (TX) and 16x12 receive (RX) FIFOs to reduce CPU interrupt service loading
 - Programmable baud-rate generator with fractional divider
 - Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
 - FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
 - Standard asynchronous communication bits for start, stop, and parity
 - False-start-bit detection
 - Line-break generation and detection
- ADC
 - Single- and differential-input configurations
 - Eight 10-bit channels (inputs) when used as single-ended inputs
 - Sample rate of one million samples/second
 - Flexible, configurable analog-to-digital conversion
 - Four programmable sample conversion sequences from one to eight entries long, with corresponding conversion result FIFOs
 - Each sequence triggered by software or internal event (timers, analog comparators, or GPIO)
 - On-chip temperature sensor
- Analog Comparators
 - Three independent integrated analog comparators
 - Configurable for output to: drive an output pin, generate an interrupt, or initiate an ADC sample sequence
 - Compare external pin input to external pin input or to internal programmable voltage reference

- I²C
 - Two I²C modules
 - Master and slave receive and transmit operation with transmission speed up to 100 Kbps in Standard mode and 400 Kbps in Fast mode
 - Interrupt generation
 - Master with arbitration and clock synchronization, multimaster support, and 7-bit addressing mode
- GPIOs
 - 3-38 GPIOs, depending on configuration
 - 5-V-tolerant input/outputs
 - Programmable interrupt generation as either edge-triggered or level-sensitive
 - Bit masking in both read and write operations through address lines
 - Can initiate an ADC sample sequence
 - Programmable control for GPIO pad configuration:
 - Weak pull-up or pull-down resistors
 - 2-mA, 4-mA, and 8-mA pad drive
 - Slew rate control for the 8-mA drive
 - Open drain enables
 - Digital input enables
- Power
 - On-chip Low Drop-Out (LDO) voltage regulator, with programmable output user-adjustable from 2.25 V to 2.75 V
 - Hibernation module handles the power-up/down 3.3 V sequencing and control for the core digital logic and analog circuits
 - Low-power options on controller: Sleep and Deep-sleep modes
 - Low-power options for peripherals: software controls shutdown of individual peripherals
 - User-enabled LDO unregulated voltage detection and automatic reset
 - 3.3-V supply brown-out detection and reporting via interrupt or reset
- Flexible Reset Sources
 - Power-on reset (POR)

- Reset pin assertion
- Brown-out (BOR) detector alerts to system power drops
- Software reset
- Watchdog timer reset
- Internal low drop-out (LDO) regulator output goes unregulated
- Additional Features
 - Six reset sources
 - Programmable clock source control
 - Clock gating to individual peripherals for power savings
 - IEEE 1149.1-1990 compliant Test Access Port (TAP) controller
 - Debug access via JTAG and Serial Wire interfaces
 - Full JTAG boundary scan
- Industrial-range 100-pin RoHS-compliant LQFP package

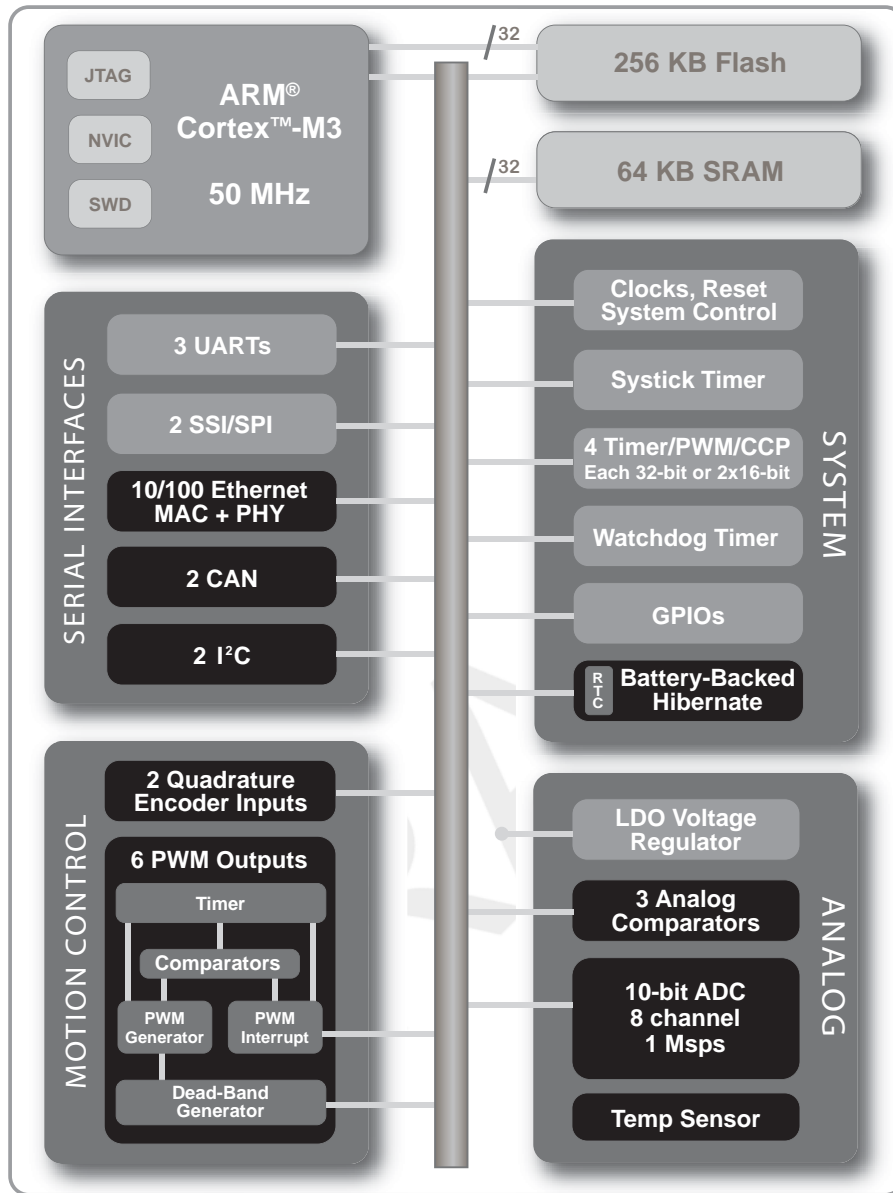
1.2 Target Applications

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

1.3 High-Level Block Diagram

Figure 1-1 on page 28 shows the features on the Stellaris® Fury-class family of devices.

Figure 1-1. Stellaris® Fury-class High-Level Block Diagram



1.4 Functional Overview

The following sections provide an overview of the features of the LM3S8938 microcontroller. The page number in parenthesis indicates where that feature is discussed in detail. Ordering and support information can be found in “Ordering and Contact Information” on page 527.

1.4.1 ARM Cortex™-M3

1.4.1.1 Processor Core (see page 35)

All members of the Stellaris® product family, including the LM3S8938 microcontroller, are designed around an ARM Cortex™-M3 processor core. The ARM Cortex-M3 processor provides the core for a high-performance, low-cost platform that meets the needs of minimal memory implementation, reduced pin count, and low-power consumption, while delivering outstanding computational performance and exceptional system response to interrupts.

“ARM Cortex-M3 Processor Core” on page 35 provides an overview of the ARM core; the core is detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

1.4.1.2 System Timer (SysTick)

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

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

1.4.1.3 Nested Vectored Interrupt Controller (NVIC)

The LM3S8938 controller includes the ARM Nested Vectored Interrupt Controller (NVIC) on the ARM Cortex-M3 core. The NVIC and Cortex-M3 prioritize and handle all exceptions. All exceptions are handled in Handler Mode. The processor state is automatically stored to the stack on an exception, and automatically restored from the stack at the end of the Interrupt Service Routine (ISR). The vector is fetched in parallel to the state saving, which enables efficient interrupt entry. The processor supports tail-chaining, which enables back-to-back interrupts to be performed without the overhead of state saving and restoration. Software can set eight priority levels on 7 exceptions (system handlers) and 34 interrupts.

“Interrupts” on page 43 provides an overview of the NVIC controller and the interrupt map. Exceptions and interrupts are detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

1.4.2 Motor Control Peripherals

To enhance motor control, the LM3S8938 controller features Pulse Width Modulation (PWM) outputs.

1.4.2.1 PWM (see page 202)

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

On the LM3S8938, PWM motion control functionality can be achieved through the motion control features of the general-purpose timers (using the CCP pins).

CCP Pins (see page 202)

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

1.4.3 Serial Communications Peripherals

The LM3S8938 controller supports both asynchronous and synchronous serial communications with:

- Three fully programmable 16C550-type UARTs
- One SSI module
- Two I²C modules
- One CAN unit

1.4.3.1 UART (see page 285)

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 LM3S8938 controller includes three fully programmable 16C550-type UARTs that support data transfer speeds up to 460.8 Kbps. In addition, each UART is capable of supporting IrDA. (Although similar in functionality to a 16C550 UART, it is not register-compatible.)

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

1.4.3.2 SSI (see page 325)

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

The LM3S8938 controller includes one SSI module that provides the functionality for synchronous serial communications with peripheral devices, and can be configured to use the Freescale SPI, MICROWIRE, or TI synchronous serial interface frame formats. The size of the data frame is also configurable, and can be set between 4 and 16 bits, inclusive.

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

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

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

1.4.3.3 I²C (see page 359)

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.

The LM3S8938 controller includes two I²C modules that provide the ability to communicate to other IC devices over an I²C bus. The I²C bus supports devices that can both transmit and receive (write and read) data.

Devices 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 also supports the simultaneous operation as both a master and a slave. The four I²C modes are: Master Transmit, Master Receive, Slave Transmit, and Slave Receive.

A Stellaris® I²C module can operate at two speeds: Standard (100 Kbps) and Fast (400 Kbps).

Both the I²C master and slave can generate interrupts. The I²C master generates interrupts when a transmit or receive operation completes (or aborts due to an error). The I²C slave generates interrupts when data has been sent or requested by a master.

1.4.3.4 Controller Area Network (see page 394)

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

A transmitter sends a message to all CAN nodes (broadcasting). Each node decides on the basis of the identifier received whether it should process the message. The identifier also determines the priority that the message enjoys in competition for bus access. Each CAN message can transmit from 0 to 8 bytes of user information. The LM3S8938 includes one CAN units.

1.4.3.5 Ethernet MAC (see page 438)

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

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

1.4.4 System Peripherals

1.4.4.1 Programmable GPIOs (see page 155)

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

The Stellaris[®] GPIO module is composed of seven physical GPIO blocks, each corresponding to an individual GPIO port. The GPIO module is FiRM-compliant (compliant to the ARM Foundation IP for Real-Time Microcontrollers specification) and supports 3-38 programmable input/output pins. The number of GPIOs available depends on the peripherals being used (see “Signal Tables” on page 496 for the signals available to each GPIO pin).

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

1.4.4.2 Four Programmable Timers (see page 196)

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

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

1.4.4.3 Watchdog Timer (see page 229)

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

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

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

1.4.5 Memory Peripherals

The LM3S8938 controller offers both SRAM and Flash memory.

1.4.5.1 SRAM (see page 131)

The LM3S8938 static random access memory (SRAM) controller supports 64 KB SRAM. The internal SRAM of the Stellaris[®] devices is located at offset 0x0000.0000 of the device memory map. To reduce the number of time-consuming read-modify-write (RMW) operations, ARM has introduced *bit-banding* technology in the new Cortex-M3 processor. With a bit-band-enabled processor, certain regions in the memory map (SRAM and peripheral space) can use address aliases to access individual bits in a single, atomic operation.

1.4.5.2 Flash (see page 132)

The LM3S8938 Flash controller supports 256 KB of flash memory. The flash is organized as a set of 1-KB blocks that can be individually erased. Erasing a block causes the entire contents of the block to be reset to all 1s. These blocks are paired into a set of 2-KB blocks that can be individually protected. The blocks can be marked as read-only or execute-only, providing different levels of code protection. Read-only blocks cannot be erased or programmed, protecting the contents of those blocks from being modified. Execute-only blocks cannot be erased or programmed, and can only

be read by the controller instruction fetch mechanism, protecting the contents of those blocks from being read by either the controller or by a debugger.

1.4.6 Additional Features

1.4.6.1 Memory Map (see page 41)

A memory map lists the location of instructions and data in memory. The memory map for the LM3S8938 controller can be found in “Memory Map” on page 41. Register addresses are given as a hexadecimal increment, relative to the module’s base address as shown in the memory map.

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

1.4.6.2 JTAG TAP Controller (see page 46)

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

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

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

1.4.6.3 System Control and Clocks (see page 57)

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

1.4.6.4 Hibernation Module (see page 113)

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

1.4.7 Hardware Details

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

- “Pin Diagram” on page 495
- “Signal Tables” on page 496

- “Operating Characteristics” on page 510
- “Electrical Characteristics” on page 511
- “Package Information” on page 525

DRAFT

2 ARM Cortex-M3 Processor Core

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

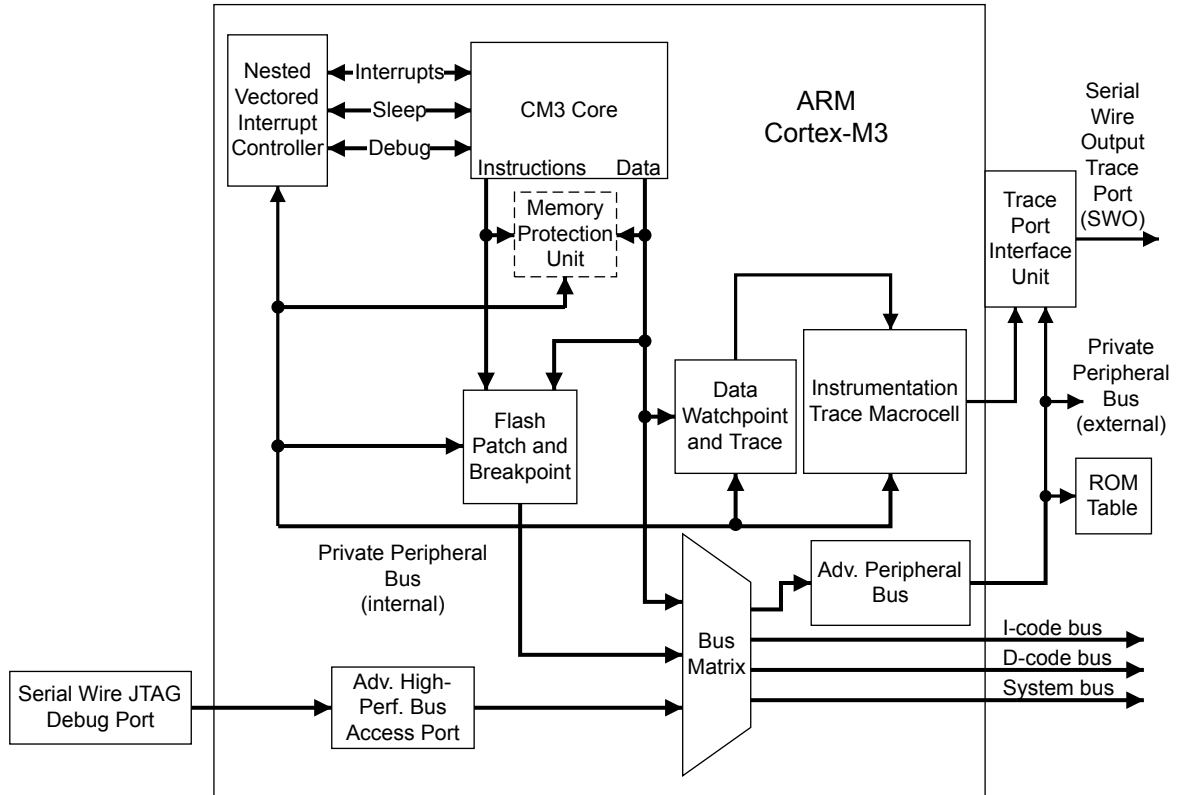
- Compact core.
- Thumb-2 instruction set, delivering the high-performance expected of an ARM core in the memory size usually associated with 8- and 16-bit devices; typically in the range of a few kilobytes of memory for microcontroller class applications.
- Speedy application execution through Harvard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

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

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

2.1 Block Diagram

Figure 2-1. CPU Block Diagram



2.2 Functional Description

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

Luminary Micro has implemented the ARM Cortex-M3 core as shown in Figure 2-1 on page 36. As noted in the *ARM® Cortex™-M3 Technical Reference Manual*, several Cortex-M3 components are flexible in their implementation: SW/JTAG-DP, ETM, TPIU, the ROM table, the MPU, and the Nested Vectored Interrupt Controller (NVIC). Each of these is addressed in the sections that follow.

2.2.1 Serial Wire and JTAG Debug

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

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

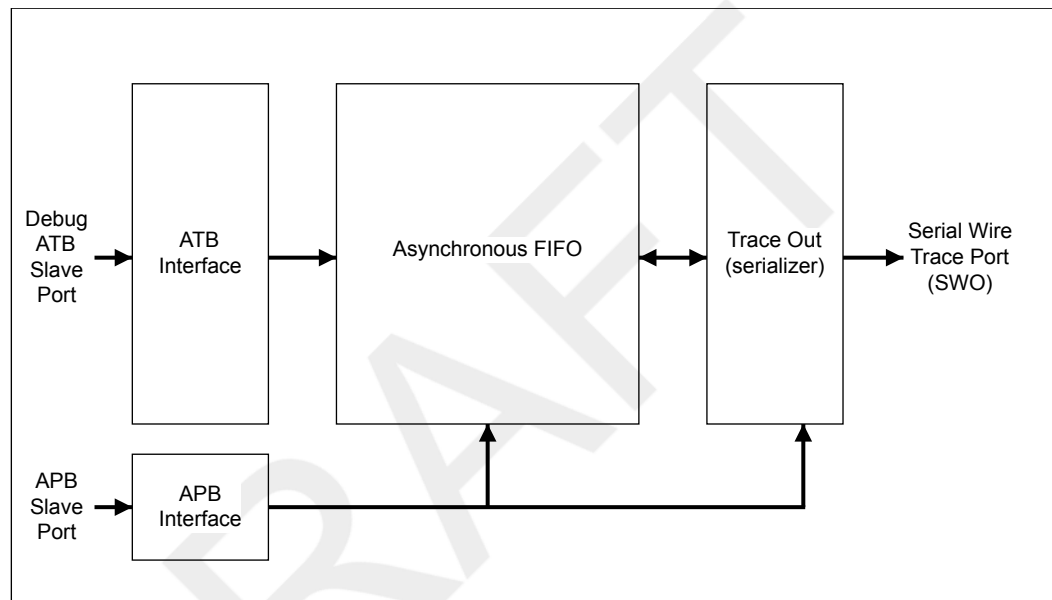
2.2.2 Embedded Trace Macrocell (ETM)

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

2.2.3 Trace Port Interface Unit (TPIU)

The TPIU acts as a bridge between the Cortex-M3 trace data from the ITM, and an off-chip Trace Port Analyzer. The Stellaris[®] devices have implemented TPIU as shown in Figure 2-2 on page 37. This is similar to the non-ETM version described in the *ARM[®] Cortex[™]-M3 Technical Reference Manual*, however, SWJ-DP only provides SWV output for the TPIU.

Figure 2-2. TPIU Block Diagram



2.2.4 ROM Table

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

2.2.5 Memory Protection Unit (MPU)

The Memory Protection Unit (MPU) is included on the LM3S8938 controller and supports the standard ARMv7 Protected Memory System Architecture (PMSA) model. The MPU provides full support for protection regions, overlapping protection regions, access permissions, and exporting memory attributes to the system.

2.2.6 Nested Vectored Interrupt Controller (NVIC)

The Nested Vectored Interrupt Controller (NVIC):

- Facilitates low-latency exception and interrupt handling

- Controls power management
- Implements system control registers

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

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

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

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

2.2.6.1 Interrupts

The *ARM® Cortex™-M3 Technical Reference Manual* describes the maximum number of interrupts and interrupt priorities. The LM3S8938 microcontroller supports 34 interrupts with eight priority levels.

2.2.6.2 System Timer (SysTick)

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

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

Functional Description

The timer consists of three registers:

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

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

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

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

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

SysTick Control and Status Register

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

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

SysTick Reload Value Register

Use the SysTick Reload Value Register to specify the start value to load into the current value register when the counter reaches 0. It can be any value between 1 and 0x00FFFFFF. A start value of 0 is possible, but has no effect because the SysTick interrupt and COUNTFLAG are activated when counting from 1 to 0.

Therefore, as a multi-shot timer, repeated over and over, it fires every N+1 clock pulse, where N is any value from 1 to 0x00FFFFFF. So, if the tick interrupt is required every 100 clock pulses, 99 must be written into the RELOAD. If a new value is written on each tick interrupt, so treated as single shot, then the actual count down must be written. For example, if a tick is next required after 400 clock pulses, 400 must be written into the RELOAD.

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

SysTick Current Value Register

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

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

SysTick Calibration Value Register

The SysTick Calibration Value register is not implemented.

3 Memory Map

The memory map for the LM3S8938 controller is provided in Table 3-1 on page 41.

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

Note: In Table 3-1 on page 41 addresses not listed are reserved.

Table 3-1. Memory Map^a

Start	End	Description	For details on registers, see page ...
Memory			
0x0000.0000	0x1FFF.FFFF	On-chip flash ^b	135
0x2000.0000	0x200F.FFFF	Bit-banded on-chip SRAM ^c	135
0x2010.0000	0x21FF.FFFF	Reserved non-bit-banded SRAM space	-
0x2200.0000	0x23FF.FFFF	Bit-band alias of 0x2000.0000 through 0x200F.FFFF	131
0x2400.0000	0x3FFF.FFFF	Reserved non-bit-banded SRAM space	-
FIRM Peripherals			
0x4000.0000	0x4000.0FFF	Watchdog timer	231
0x4000.1000	0x4000.3FFF	Reserved	-
0x4000.4000	0x4000.4FFF	GPIO Port A	161
0x4000.5000	0x4000.5FFF	GPIO Port B	161
0x4000.6000	0x4000.6FFF	GPIO Port C	161
0x4000.7000	0x4000.7FFF	GPIO Port D	161
0x4000.8000	0x4000.8FFF	SSI0	336
0x4000.A000	0x4000.BFFF	Reserved	-
0x4000.C000	0x4000.CFFF	UART0	292
0x4000.D000	0x4000.DFFF	UART1	292
0x4000.E000	0x4000.EFFF	UART2	292
0x4000.F000	0x4000.FFFF	Reserved	-
0x4001.0000	0x4001.FFFF	Reserved for future FiRM peripherals	-
Peripherals			
0x4002.0000	0x4002.07FF	I2C Master 0	372
0x4002.0800	0x4002.0FFF	I2C Slave 0	385
0x4002.1000	0x4002.17FF	I2C Master 1	372
0x4001.1800	0x4002.1FFF	I2C Slave 1	385
0x4002.2000	0x4002.3FFF	Reserved	-
0x4002.4000	0x4002.4FFF	GPIO Port E	161
0x4002.5000	0x4002.5FFF	GPIO Port F	161
0x4002.6000	0x4002.6FFF	GPIO Port G	161
0x4002.9000	0x4002.BFFF	Reserved	-
0x4002.E000	0x4002.FFFF	Reserved	-

Start	End	Description	For details on registers, see page ...
0x4003.0000	0x4003.0FFF	Timer0	207
0x4003.1000	0x4003.1FFF	Timer1	207
0x4003.2000	0x4003.2FFF	Timer2	207
0x4003.3000	0x4003.3FFF	Timer3	207
0x4003.4000	0x4003.7FFF	Reserved	-
0x4003.8000	0x4003.8FFF	ADC	258
0x4003.9000	0x4003.BFFF	Reserved	-
0x4003.C000	0x4003.CFFF	Analog Comparators	482
0x4003.D000	0x4003.FFFF	Reserved	-
0x4004.0000	0x4004.0FFF	CAN0 Controller	407
0x4004.3000	0x4004.7FFF	Reserved	-
0x4004.8000	0x4004.8FFF	Ethernet Controller	446
0x4004.9000	0x4004.BFFF	Reserved	-
0x4004.C000	0x400F.BFFF	Reserved	-
0x400F.C000	0x400F.CFFF	Hibernation Module	118
0x400F.D000	0x400F.DFFF	Flash control	135
0x400F.E000	0x400F.EFFF	System control	64
0x400F.F000	0x400F.FFFF	Reserved	-
0x4011.1000	0x4011.1FFF	Reserved	-
0x4012.0000	0x41FF.FFFF	Reserved for non bit-banded peripheral space	-
0x4200.0000	0x43FF.FFFF	Bit-banded alias of 0x4000.0000 through 0x400F.FFFF	-
0x4400.0000	0x5E32.FFFF	Reserved for non bit-banded peripheral space	-
0x5E34.0000	0x5FFF.FFFF	Reserved	-
0x6000.0000	0xDFFF.FFFF	Reserved for external devices	-
Private Peripheral Bus			
0xE000.0000	0xE000.0FFF	Instrumentation Trace Macrocell (ITM)	ARM® Cortex™-M3 Technical Reference Manual
0xE000.1000	0xE000.1FFF	Data Watchpoint and Trace (DWT)	
0xE000.2000	0xE000.2FFF	Flash Patch and Breakpoint (FPB)	
0xE000.3000	0xE000.DFFF	Reserved	
0xE000.E000	0xE000.EFFF	Nested Vectored Interrupt Controller (NVIC)	
0xE000.F000	0xE003.FFFF	Reserved	
0xE004.0000	0xE004.0FFF	Trace Port Interface Unit (TPIU)	
0xE004.1000	0xE004.1FFF	Reserved	-
0xE004.2000	0xE00F.FFFF	Reserved	-
0xE010.0000	0xFFFF.FFFF	Reserved for vendor peripherals	-

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

4 Interrupts

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

Table 4-1 on page 43 lists all the exceptions. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 34 interrupts (listed in Table 4-2 on page 44).

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

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

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

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

Note: In Table 4-2 on page 44 interrupts not listed are reserved.

Table 4-1. Exception Types

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

Exception Type	Position	Priority ^a	Description
Debug Monitor	12	settable	Debug monitor (when not halting). This is synchronous, but only active when enabled. It does not activate if lower priority than the current activation.
-	13	-	Reserved.
PendSV	14	settable	Pendable request for system service. This is asynchronous and only pended by software.
SysTick	15	settable	System tick timer has fired. This is asynchronous.
Interrupts	16 and above	settable	Asserted from outside the ARM Cortex-M3 core and fed through the NVIC (prioritized). These are all asynchronous. Table 4-2 on page 44 lists the interrupts on the LM3S8938 controller.

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

Table 4-2. Interrupts

Interrupt (Bit in Interrupt Registers)	Description
0	GPIO Port A
1	GPIO Port B
2	GPIO Port C
3	GPIO Port D
4	GPIO Port E
5	UART0
6	UART1
7	SSI0
8	I2C0
14	ADC Sequence 0
15	ADC Sequence 1
16	ADC Sequence 2
17	ADC Sequence 3
18	Watchdog timer
19	Timer0 A
20	Timer0 B
21	Timer1 A
22	Timer1 B
23	Timer2 A
24	Timer2 B
25	Analog Comparator 0
26	Analog Comparator 1
27	Analog Comparator 2
28	System Control
29	Flash Control
30	GPIO Port F
31	GPIO Port G
33	UART2
35	Timer3 A
36	Timer3 B

Interrupt (Bit in Interrupt Registers)	Description
37	I2C1
39	CAN0
42	Ethernet Controller
43	Hibernation Module
44-47	Reserved

DRAFT

5 JTAG Interface

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

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

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

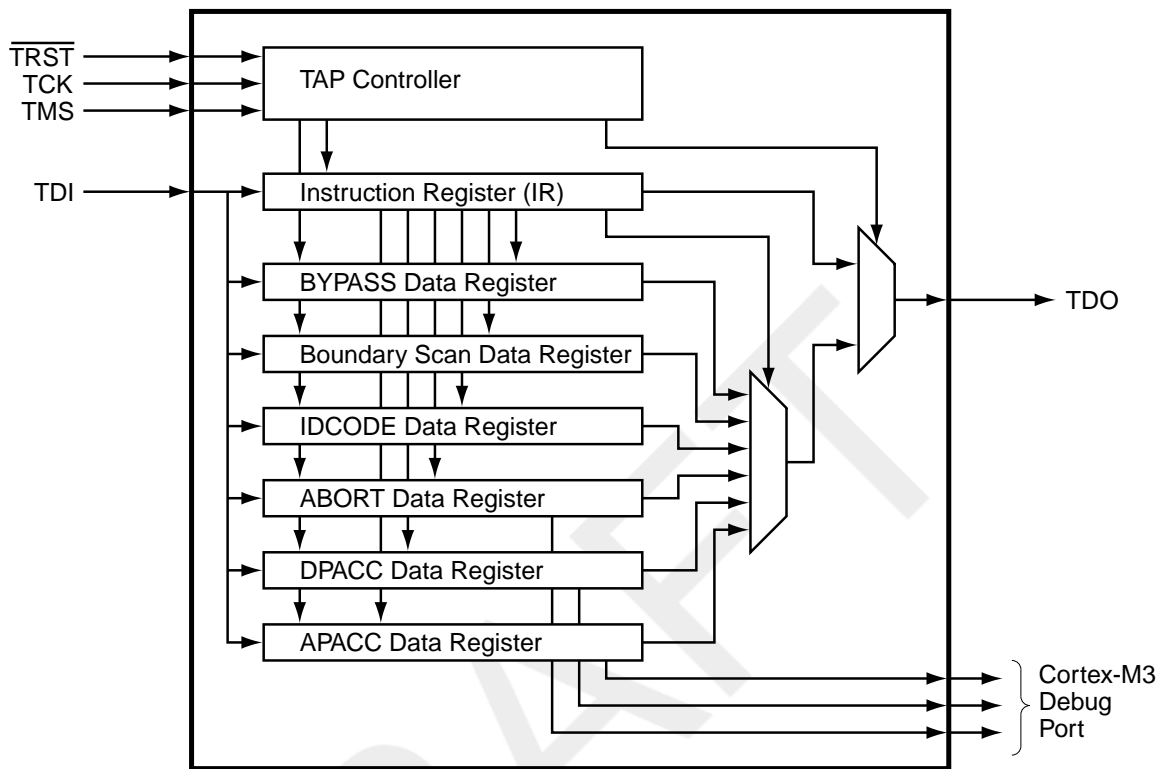
The JTAG module has the following features:

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

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

5.1 Block Diagram

Figure 5-1. JTAG Module Block Diagram



5.2 Functional Description

A high-level conceptual drawing of the JTAG module is shown in Figure 5-1 on page 47. The JTAG module is composed of the Test Access Port (TAP) controller and serial shift chains with parallel update registers. The TAP controller is a simple state machine controlled by the $\overline{\text{TRST}}$, TCK and TMS inputs. The current state of the TAP controller depends on the current value of $\overline{\text{TRST}}$ and the sequence of values captured on TMS at the rising edge of TCK. The TAP controller determines when the serial shift chains capture new data, shift data from TDI towards TDO, and update the parallel load registers. The current state of the TAP controller also determines whether the Instruction Register (IR) chain or one of the Data Register (DR) chains is being accessed.

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

Some instructions, like EXTEST and INTEST, operate on data currently in a DR chain and do not capture, shift, or update any of the chains. Instructions that are not implemented decode to the BYPASS instruction to ensure that the serial path between TDI and TDO is always connected (see Table 5-2 on page 53 for a list of implemented instructions).

See “JTAG and Boundary Scan” on page 521 for JTAG timing diagrams.

5.2.1 JTAG Interface Pins

The JTAG interface consists of five standard pins: $\overline{\text{TRST}}$, TCK, TMS, TDI, and TDO. These pins and their associated reset state are given in Table 5-1 on page 48. Detailed information on each pin follows.

Table 5-1. JTAG Port Pins Reset State

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

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

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

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

5.2.1.2 Test Clock Input (TCK)

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

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

5.2.1.3 Test Mode Select (TMS)

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

Holding TMS high for five consecutive TCK cycles drives the TAP controller state machine to the Test-Logic-Reset state. When the TAP controller enters the Test-Logic-Reset state, the JTAG Instruction Register (IR) resets to the default instruction, IDCODE. Therefore, this sequence can be used as a reset mechanism, similar to asserting $\overline{\text{TRST}}$. The JTAG Test Access Port state machine can be seen in its entirety in Figure 5-2 on page 50.

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

5.2.1.4 Test Data Input (TDI)

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

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

5.2.1.5 Test Data Output (TDO)

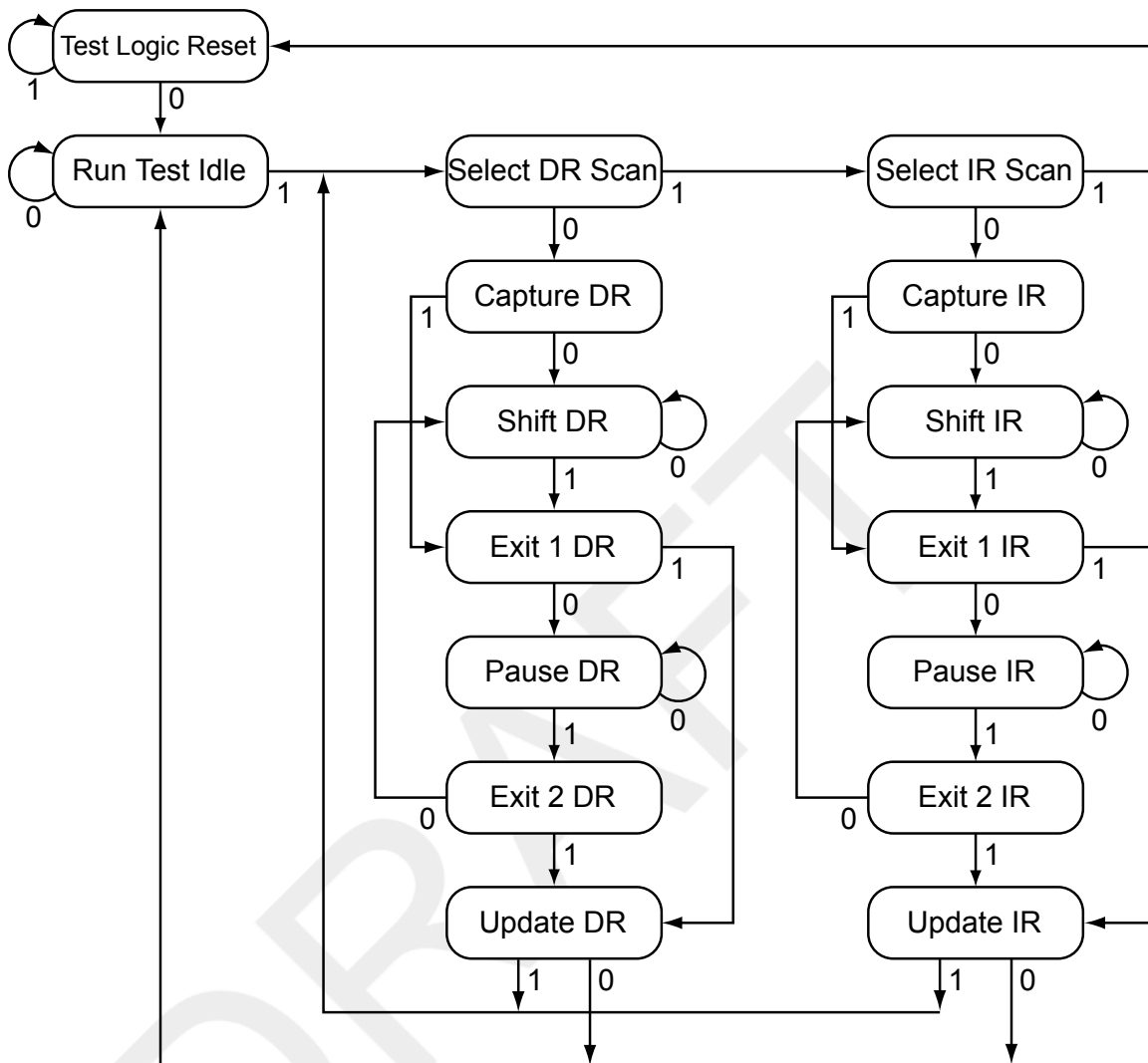
The TDO pin provides an output stream of serial information from the IR chain or the DR chains. The value of TDO depends on the current TAP state, the current instruction, and the data in the chain being accessed. In order to save power when the JTAG port is not being used, the TDO pin is placed in an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

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

5.2.2 JTAG TAP Controller

The JTAG TAP controller state machine is shown in Figure 5-2 on page 50. The TAP controller state machine is reset to the Test-Logic-Reset state on the assertion of a Power-On-Reset (POR) or the assertion of TRST. Asserting the correct sequence on the TMS pin allows the JTAG module to shift in new instructions, shift in data, or idle during extended testing sequences. For detailed information on the function of the TAP controller and the operations that occur in each state, please refer to *IEEE Standard 1149.1*.

Figure 5-2. Test Access Port State Machine



5.2.3 Shift Registers

The Shift Registers consist of a serial shift register chain and a parallel load register. The serial shift register chain samples specific information during the TAP controller's CAPTURE states and allows this information to be shifted out of TDO during the TAP controller's SHIFT states. While the sampled data is being shifted out of the chain on TDO, new data is being shifted into the serial shift register on TDI. This new data is stored in the parallel load register during the TAP controller's UPDATE states. Each of the shift registers is discussed in detail in "Register Descriptions" on page 53.

5.2.4 Operational Considerations

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

5.2.4.1 GPIO Functionality

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

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

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

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

The commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Writes to protected bits of the **GPIO Alternate Function Select (GPIO AFSEL)** register (see page 171) are not committed to storage unless the **GPIO Lock (GPIO LOCK)** register (see page 181) has been unlocked and the appropriate bits of the **GPIO Commit (GPIO CR)** register (see page 182) have been set to 1.

Recovering a "Locked" Device

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

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

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

The JTAG-to-SWD and SWD-to-JTAG switch sequences are described in “ARM Serial Wire Debug (SWD)” on page 52. When performing switch sequences for the purpose of recovering the debug capabilities of the device, only steps 1 and 2 of the switch sequence need to be performed.

5.2.4.2 ARM Serial Wire Debug (SWD)

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

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

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

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

JTAG-to-SWD Switching

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

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

SWD-to-JTAG Switching

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

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

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

5.3 Initialization and Configuration

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

5.4 Register Descriptions

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

5.4.1 Instruction Register (IR)

The JTAG TAP Instruction Register (IR) is a four-bit serial scan chain with a parallel load register connected between the JTAG TDI and TDO pins. When the TAP Controller is placed in the correct states, bits can be shifted into the Instruction Register. Once these bits have been shifted into the chain and updated, they are interpreted as the current instruction. The decode of the Instruction Register bits is shown in Table 5-2 on page 53. A detailed explanation of each instruction, along with its associated Data Register, follows.

Table 5-2. JTAG Instruction Register Commands

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

5.4.1.1 EXTEST Instruction

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

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

5.4.1.2 INTEST Instruction

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

5.4.1.3 SAMPLE/PRELOAD Instruction

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

While these samples of the inputs, outputs, and output enables are being shifted out of the Boundary Scan Data Register, new data is being shifted into the Boundary Scan Data Register from TDI. Once the new data has been shifted into the Boundary Scan Data Register, the data is saved in the parallel load registers when the TAP controller enters the Update DR state. This update of the parallel load register preloads data into the Boundary Scan Data Register that is associated with each input, output, and output enable. This preloaded data can be used with the EXTEST and INTEST instructions to drive data into or out of the controller. Please see “Boundary Scan Data Register” on page 56 for more information.

5.4.1.4 ABORT Instruction

The ABORT instruction connects the associated ABORT Data Register chain between TDI and TDO. This instruction provides read and write access to the ABORT Register of the ARM Debug Access Port (DAP). Shifting the proper data into this Data Register clears various error bits or initiates a DAP abort of a previous request. Please see the “ABORT Data Register” on page 56 for more information.

5.4.1.5 DPACC Instruction

The DPACC instruction connects the associated DPACC Data Register chain between TDI and TDO. This instruction provides read and write access to the DPACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to the ARM debug and status registers. Please see “DPACC Data Register” on page 56 for more information.

5.4.1.6 APACC Instruction

The APACC instruction connects the associated APACC Data Register chain between TDI and TDO. This instruction provides read and write access to the APACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to internal components and buses through the Debug Port. Please see “APACC Data Register” on page 56 for more information.

5.4.1.7 IDCODE Instruction

The IDCODE instruction connects the associated IDCODE Data Register chain between TDI and TDO. This instruction provides information on the manufacturer, part number, and version of the ARM core. This information can be used by testing equipment and debuggers to automatically configure their input and output data streams. IDCODE is the default instruction that is loaded into the JTAG Instruction Register when a power-on-reset (POR) is asserted, $\overline{\text{TRST}}$ is asserted, or the Test-Logic-Reset state is entered. Please see “IDCODE Data Register” on page 55 for more information.

5.4.1.8 BYPASS Instruction

The BYPASS instruction connects the associated BYPASS Data Register chain between TDI and TDO. This instruction is used to create a minimum length serial path between the TDI and TDO ports. The BYPASS Data Register is a single-bit shift register. This instruction improves test efficiency by allowing components that are not needed for a specific test to be bypassed in the JTAG scan chain by loading them with the BYPASS instruction. Please see “BYPASS Data Register” on page 55 for more information.

5.4.2 Data Registers

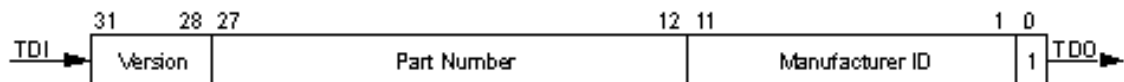
The JTAG module contains six Data Registers. These include: IDCODE, BYPASS, Boundary Scan, APACC, DPACC, and ABORT serial Data Register chains. Each of these Data Registers is discussed in the following sections.

5.4.2.1 IDCODE Data Register

The format for the 32-bit IDCODE Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 5-3 on page 55. The standard requires that every JTAG-compliant device implement either the IDCODE instruction or the BYPASS instruction as the default instruction. The LSB of the IDCODE Data Register is defined to be a 1 to distinguish it from the BYPASS instruction, which has an LSB of 0. This allows auto configuration test tools to determine which instruction is the default instruction.

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

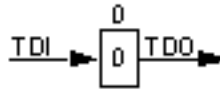
Figure 5-3. IDCODE Register Format



5.4.2.2 BYPASS Data Register

The format for the 1-bit BYPASS Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 5-4 on page 56. The standard requires that every JTAG-compliant device implement either the BYPASS instruction or the IDCODE instruction as the default instruction. The LSB of the BYPASS Data Register is defined to be a 0 to distinguish it from the IDCODE instruction, which has an LSB of 1. This allows auto configuration test tools to determine which instruction is the default instruction.

Figure 5-4. BYPASS Register Format

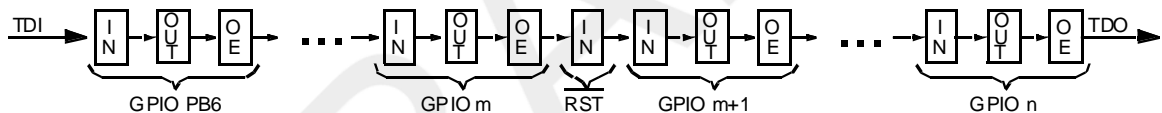


5.4.2.3 Boundary Scan Data Register

The format of the Boundary Scan Data Register is shown in Figure 5-5 on page 56. Each GPIO pin, in a counter-clockwise direction from the JTAG port pins, is included in the Boundary Scan Data Register. Each GPIO pin has three associated digital signals that are included in the chain. These signals are input, output, and output enable, and are arranged in that order as can be seen in the figure. In addition to the GPIO pins, the controller reset pin, \overline{RST} , is included in the chain. Because the reset pin is always an input, only the input signal is included in the Data Register chain.

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

Figure 5-5. Boundary Scan Register Format



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

5.4.2.4 APACC Data Register

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

5.4.2.5 DPACC Data Register

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

5.4.2.6 ABORT Data Register

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

6 System Control

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

6.1 Functional Description

The System Control module provides the following capabilities:

- Device identification, see “Device Identification” on page 57
- Local control, such as reset (see “Reset Control” on page 57), power (see “Power Control” on page 60) and clock control (see “Clock Control” on page 60)
- System control (Run, Sleep, and Deep-Sleep modes), see “System Control” on page 62

6.1.1 Device Identification

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

6.1.2 Reset Control

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

6.1.2.1 CMOD0 and CMOD1 Test-Mode Control Pins

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

6.1.2.2 Reset Sources

The controller has five sources of reset:

1. External reset input pin ($\overline{\text{RST}}$) assertion, see “ $\overline{\text{RST}}$ Pin Assertion” on page 57.
2. Power-on reset (POR), see “Power-On Reset (POR)” on page 58.
3. Internal brown-out (BOR) detector, see “Brown-Out Reset (BOR)” on page 58.
4. Software-initiated reset (with the software reset registers), see “Software Reset” on page 59.
5. A watchdog timer reset condition violation, see “Watchdog Timer Reset” on page 59.

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

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

The external reset pin ($\overline{\text{RST}}$) resets the controller. This resets the core and all the peripherals except the JTAG TAP controller (see “JTAG Interface” on page 46). The external reset sequence is as follows:

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

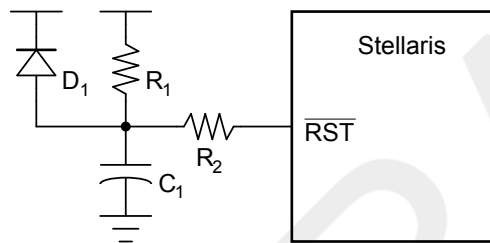
The external reset timing is shown in Figure 22-11 on page 523.

6.1.2.4 Power-On Reset (POR)

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

The device must be operating within the specified operating parameters at the point when the on-chip power-on reset pulse is complete. The 3.3-V power supply to the device must reach 3.0 V within 10 msec of it crossing 2.0 V to guarantee proper operation. For applications that require the use of an external reset to hold the device in reset longer than the internal POR, the \overline{RST} input may be used with the circuit as shown in Figure 6-1 on page 58.

Figure 6-1. External Circuitry to Extend Reset



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

The Power-On Reset sequence is as follows:

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

The internal POR is only active on the initial power-up of the controller. The Power-On Reset timing is shown in Figure 22-12 on page 524.

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

6.1.2.5 Brown-Out Reset (BOR)

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

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

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

The brown-out reset is equivalent to an assertion of the external \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 22-13 on page 524.

6.1.2.6 Software Reset

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

Peripherals can be individually reset by software via three registers that control reset signals to each peripheral (see the **SRCRn** registers). If the bit position corresponding to a peripheral is set, 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 62). Writing a bit lane with a value of 1 initiates a reset of the corresponding unit. Note that all reset signals for all clocks of the specified unit are asserted as a result of a software-initiated reset.

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

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

The software-initiated system reset timing is shown in Figure 22-14 on page 524.

6.1.2.7 Watchdog Timer Reset

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

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

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

The watchdog reset timing is shown in Figure 22-15 on page 524.

6.1.3 Power Control

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

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

6.1.4 Clock Control

System control determines the control of clocks in this part.

6.1.4.1 Fundamental Clock Sources

There are four clock sources for use in the device:

- **Internal Oscillator (IOSC):** The internal oscillator is an on-chip clock source. It does not require the use of any external components. The frequency of the internal oscillator is $12\text{ MHz} \pm 30\%$. Applications that do not depend on accurate clock sources may use this clock source to reduce system cost. The internal oscillator is the clock source the device uses during and following POR. If the main oscillator is required, software must enable the main oscillator following reset and allow the main oscillator to stabilize before changing the clock reference.
- **Main Oscillator:** The main oscillator provides a frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the OSC0 input pin, or an external crystal is connected across the OSC0 input and OSC1 output pins. The crystal value allowed depends on whether the main oscillator is used as the clock reference source to the PLL. If so, the crystal must be one of the supported frequencies between 3.579545 MHz through 8.192 MHz (inclusive). If the PLL is not being used, the crystal may be any one of the supported frequencies between 1 MHz and 8.192 MHz. The single-ended clock source range is from DC through the specified speed of the device. The supported crystals are listed in Table 6-3 on page 75.
- **Internal 30-kHz oscillator:** The internal 30-kHz oscillator is similar to the internal oscillator, except that it provides an operational frequency of $30\text{ kHz} \pm 30\%$. It is intended for use during Deep-Sleep power-saving modes. This power-savings mode benefits from reduced internal switching and also allows the main oscillator to be powered down.
- **External real-time oscillator:** The external real-time oscillator provides a low-frequency, accurate clock reference. It is intended to provide the system with a real-time clock source. The real-time oscillator is part of the Hibernation Module (“Hibernation Module” on page 113) and may also provide an accurate source of Deep-Sleep or Hibernate mode power savings.

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

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

6.1.4.2 Crystal Configuration for the Main Oscillator (MOSC)

The main oscillator supports the use of a select number of crystals in the range of 1 MHz through 8.192 MHz. This method allows Luminary Micro to provide the best possible PLL settings.

Table 6-3 on page 75 describes the available crystal choices and default programming values.

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

6.1.4.3 PLL Frequency Configuration

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

If the main oscillator provides the clock reference to the PLL, the translation provided by hardware and used to program the PLL is available for software in the **XTAL to PLL Translation (PLLCFG)** register (see page 77). The internal translation provides a translation within $\pm 1\%$ of the targetted PLL VCO frequency.

Table 6-3 on page 75 describes the available crystal choices and default programming of the **PLLCFG** register. The crystal number is written into the `XTAL` field of the **Run-Mode Clock Configuration (RCC)** register. Any time the `XTAL` field changes, the new settings are translated and the internal PLL settings are updated.

6.1.4.4 PLL Modes

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

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

The modes are programmed using the **RCC/RCC2** register fields (see page 73 and page 78).

6.1.4.5 PLL Operation

If the PLL configuration is changed, the PLL output frequency is unstable until it reconverges (relocks) to the new setting. The time between the configuration change and relock is T_{READY} (see Table 22-5 on page 513). During this time, the PLL is not usable as a clock reference.

The PLL is changed by one of the following:

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

A counter is defined to measure the T_{READY} requirement. The counter is clocked by the main oscillator. The range of the main oscillator has been taken into account and the down counter is set to 0x1200 (that is, $\sim 600 \mu\text{s}$ at a 8.192 MHz external oscillator clock). Hardware is provided to keep the PLL from being used as a system clock until the T_{READY} condition is met after one of the two

changes above. It is the user's responsibility to have a stable clock source (like the main oscillator) before the **RCC/RCC2** register is switched to use the PLL.

6.1.5 System Control

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

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

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

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

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

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

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

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

code. It can determine that it has been restarted from Hibernate mode by inspecting the Hibernation module registers.

6.2 Initialization and Configuration

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

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

6.3 Register Map

Table 6-1 on page 63 lists the System Control registers, grouped by function. The offset listed is a hexadecimal increment to the register’s address, relative to the System Control base address of 0x400F.E000.

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

Note: A BV in the Reset column indicates the reset value is a Build Value and part-specific. See the page number referenced for the reset value description.

Table 6-1. System Control Register Map

Offset	Name	Type	Reset	Description	See page
0x000	DID0	RO	-	Device Identification 0	65
0x004	DID1	RO	-	Device Identification 1	81
0x008	DC0	RO	0x00FF.007F	Device Capabilities 0	83
0x010	DC1	RO	0x0101.33FF	Device Capabilities 1	84
0x014	DC2	RO	0x070F.5017	Device Capabilities 2	86
0x018	DC3	RO	0x3FFF.7FC0	Device Capabilities 3	88
0x01C	DC4	RO	0x5100.007F	Device Capabilities 4	90
0x030	PBORCTL	R/W	0x0000.7FFD	Brown-Out Reset Control	67

Offset	Name	Type	Reset	Description	See page
0x034	LDOPCTL	R/W	0x0000.0000	LDO Power Control	68
0x040	SRCR0	R/W	0x00000000	Software Reset Control 0	109
0x044	SRCR1	R/W	0x00000000	Software Reset Control 1	110
0x048	SRCR2	R/W	0x00000000	Software Reset Control 2	112
0x050	RIS	RO	0x0000.0000	Raw Interrupt Status	69
0x054	IMC	R/W	0x0000.0000	Interrupt Mask Control	70
0x058	MISC	R/W1C	0x0000.0000	Masked Interrupt Status and Clear	71
0x05C	RESC	R/W	-	Reset Cause	72
0x060	RCC	R/W	0x07A0.3AD1	Run-Mode Clock Configuration	73
0x064	PLLCFG	RO	-	XTAL to PLL Translation	77
0x070	RCC2	R/W	0x0780.2800	Run-Mode Clock Configuration 2	78
0x100	RCGC0	R/W	0x00000040	Run Mode Clock Gating Control Register 0	91
0x104	RCGC1	R/W	0x00000000	Run Mode Clock Gating Control Register 1	97
0x108	RCGC2	R/W	0x00000000	Run Mode Clock Gating Control Register 2	103
0x110	SCGC0	R/W	0x00000040	Sleep Mode Clock Gating Control Register 0	93
0x114	SCGC1	R/W	0x00000000	Sleep Mode Clock Gating Control Register 1	99
0x118	SCGC2	R/W	0x00000000	Sleep Mode Clock Gating Control Register 2	105
0x120	DCGC0	R/W	0x00000040	Deep Sleep Mode Clock Gating Control Register 0	95
0x124	DCGC1	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 1	101
0x128	DCGC2	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 2	107
0x144	DSLPCCLKCFG	R/W	0x0780.0000	Deep Sleep Clock Configuration	80

6.4 Register Descriptions

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

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

This register identifies the version of the device.

Device Identification 0 (DID0)

Base 0x400F.E000

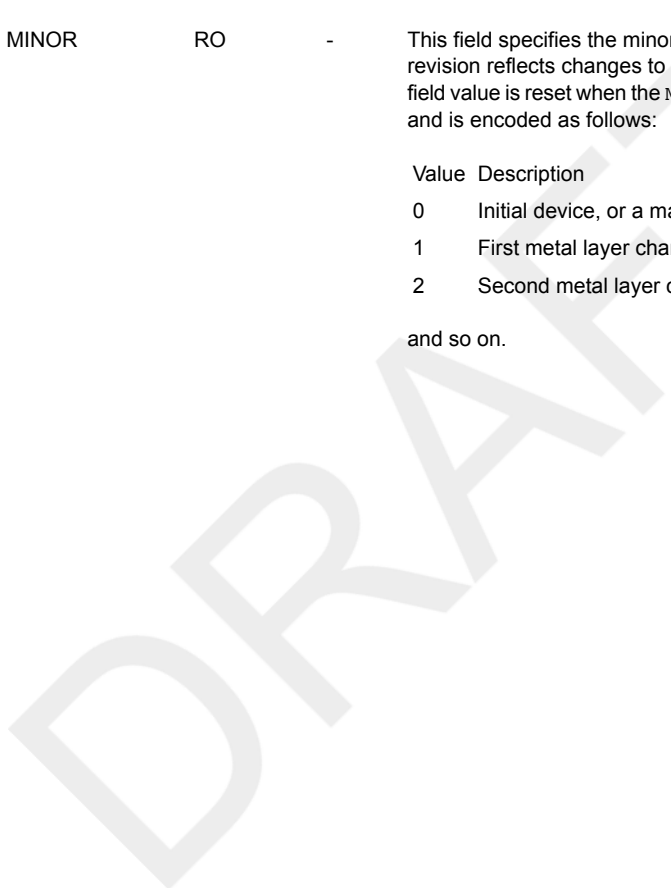
Offset 0x000

Type RO, reset -

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

Bit/Field	Name	Type	Reset	Description						
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.						
30:28	VER	RO	1	This field defines the DID0 register format version. The version number is numeric. The value of the VER field is encoded as follows: <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>First revision of the DID0 register format, for Stellaris® Fury-class devices.</td> </tr> </tbody> </table>	Value	Description	1	First revision of the DID0 register format, for Stellaris® Fury-class devices.		
Value	Description									
1	First revision of the DID0 register format, for Stellaris® Fury-class devices.									
27:24	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.						
23:16	CLASS	RO	1	The CLASS field value identifies the internal design from which all mask sets are generated for all devices 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 devices. The value of the CLASS field is encoded as follows (all other encodings are reserved): <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Stellaris® Sandstorm-class devices.</td> </tr> <tr> <td>1</td> <td>Stellaris® Fury-class devices.</td> </tr> </tbody> </table>	Value	Description	0	Stellaris® Sandstorm-class devices.	1	Stellaris® Fury-class devices.
Value	Description									
0	Stellaris® Sandstorm-class devices.									
1	Stellaris® Fury-class devices.									

Bit/Field	Name	Type	Reset	Description								
15:8	MAJOR	RO	-	<p>This field specifies the major revision number of the device. The major revision reflects changes to base layers of the design. The major revision number is indicated in the part number as a letter (A for first revision, B for second, and so on). This field is encoded as follows:</p> <table border="1"><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>Revision A (initial device)</td></tr><tr><td>1</td><td>Revision B (first base layer revision)</td></tr><tr><td>2</td><td>Revision C (second base layer revision)</td></tr></tbody></table> <p>and so on.</p>	Value	Description	0	Revision A (initial device)	1	Revision B (first base layer revision)	2	Revision C (second base layer revision)
Value	Description											
0	Revision A (initial device)											
1	Revision B (first base layer revision)											
2	Revision C (second base layer revision)											
7:0	MINOR	RO	-	<p>This field specifies the minor revision number of the device. The minor revision reflects changes to the metal layers of the design. The <code>MINOR</code> field value is reset when the <code>MAJOR</code> field is changed. This field is numeric and is encoded as follows:</p> <table border="1"><thead><tr><th>Value</th><th>Description</th></tr></thead><tbody><tr><td>0</td><td>Initial device, or a major revision update.</td></tr><tr><td>1</td><td>First metal layer change.</td></tr><tr><td>2</td><td>Second metal layer change.</td></tr></tbody></table> <p>and so on.</p>	Value	Description	0	Initial device, or a major revision update.	1	First metal layer change.	2	Second metal layer change.
Value	Description											
0	Initial device, or a major revision update.											
1	First metal layer change.											
2	Second metal layer change.											



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

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

Brown-Out Reset Control (PBORCTL)

Base 0x400F.E000

Offset 0x030

Type R/W, reset 0x0000.7FFD

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

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

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

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

LDO Power Control (LDOPCTL)

Base 0x400F.E000

Offset 0x034

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	VADJ	R/W	0x0	This field sets the on-chip output voltage. The programming values for the V_{ADJ} field are provided in Table 6-2 on page 68.

Table 6-2. VADJ to VOUT

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

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

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

Raw Interrupt Status (RIS)

Base 0x400F.E000

Offset 0x050

Type RO, reset 0x0000.0000

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

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

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

Central location for system control interrupt masks.

Interrupt Mask Control (IMC)

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

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

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

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

Central location for system control result of RIS AND IMC to generate an interrupt to the controller. All of the bits are R/W1C and this action also clears the corresponding raw interrupt bit in the **RIS** register (see page 69).

Masked Interrupt Status and Clear (MISC)

Base 0x400F.E000

Offset 0x058

Type R/W1C, reset 0x0000.0000

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

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

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

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

Reset Cause (RESC)

Base 0x400F.E000

Offset 0x05C

Type R/W, reset -

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

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	LDO	R/W	-	When set, indicates the LDO circuit has lost regulation and has generated a reset event.
4	SW	R/W	-	When set, indicates a software reset is the cause of the reset event.
3	WDT	R/W	-	When set, indicates a watchdog reset is the cause of the reset event.
2	BOR	R/W	-	When set, indicates a brown-out reset is the cause of the reset event.
1	POR	R/W	-	When set, indicates a power-on reset is the cause of the reset event.
0	EXT	R/W	-	When set, indicates an external reset (\overline{RST} assertion) is the cause of the reset event.

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

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

Run-Mode Clock Configuration (RCC)

Base 0x400F.E000

Offset 0x060

Type R/W, reset 0x07A0.3AD1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved				ACG	SYSDIV				USESYSOVM	reserved					
Type	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		PWRDN	reserved	BYPASS	reserved	XTAL				OSCSRC		reserved		IOSCDIS	MOSCDIS
Type	RO	RO	R/W	RO	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	R/W	R/W
Reset	0	1	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 controller enters a Sleep or Deep-Sleep mode (respectively). If set, the SCGCn or DCGCn registers are used to control the clocks distributed to the peripherals when the controller is in a sleep mode. Otherwise, the Run-Mode Clock Gating Control (RCGCn) registers are used when the controller enters a sleep mode.</p> <p>The RCGCn registers are always used to control the clocks in Run mode.</p> <p>This allows peripherals to consume less power when the controller is in a sleep mode and the peripheral is unused.</p>

Bit/Field	Name	Type	Reset	Description																																													
26:23	SYSDIV	R/W	0xF	<p>System Clock Divisor</p> <p>Specifies which divisor is used to generate the system clock from the PLL output.</p> <p>The PLL VCO frequency is 400 MHz.</p> <table border="1"> <thead> <tr> <th>Binary Value</th> <th>Divisor (BYPASS=1)</th> <th>Frequency (BYPASS=0)</th> </tr> </thead> <tbody> <tr> <td>0000-0010</td> <td>reserved</td> <td>reserved</td> </tr> <tr> <td>0011</td> <td>/8</td> <td>50 MHz</td> </tr> <tr> <td>0100</td> <td>/10</td> <td>40 MHz</td> </tr> <tr> <td>0101</td> <td>/12</td> <td>33.33 MHz</td> </tr> <tr> <td>0110</td> <td>/14</td> <td>28.57 MHz</td> </tr> <tr> <td>0111</td> <td>/16</td> <td>25 MHz</td> </tr> <tr> <td>1000</td> <td>/18</td> <td>22.22 MHz</td> </tr> <tr> <td>1001</td> <td>/20</td> <td>20 MHz</td> </tr> <tr> <td>1010</td> <td>/22</td> <td>18.18 MHz</td> </tr> <tr> <td>1011</td> <td>/24</td> <td>16.67 MHz</td> </tr> <tr> <td>1100</td> <td>/26</td> <td>15.38 MHz</td> </tr> <tr> <td>1101</td> <td>/28</td> <td>14.29 MHz</td> </tr> <tr> <td>1110</td> <td>/30</td> <td>13.33 MHz</td> </tr> <tr> <td>1111</td> <td>/32</td> <td>12.5 MHz (default)</td> </tr> </tbody> </table> <p>When reading the Run-Mode Clock Configuration (RCC) register (see page 73), the SYSDIV value is MINSYSDIV if a lower divider was requested and the PLL is being used. This lower value is allowed to divide a non-PLL source.</p>	Binary Value	Divisor (BYPASS=1)	Frequency (BYPASS=0)	0000-0010	reserved	reserved	0011	/8	50 MHz	0100	/10	40 MHz	0101	/12	33.33 MHz	0110	/14	28.57 MHz	0111	/16	25 MHz	1000	/18	22.22 MHz	1001	/20	20 MHz	1010	/22	18.18 MHz	1011	/24	16.67 MHz	1100	/26	15.38 MHz	1101	/28	14.29 MHz	1110	/30	13.33 MHz	1111	/32	12.5 MHz (default)
Binary Value	Divisor (BYPASS=1)	Frequency (BYPASS=0)																																															
0000-0010	reserved	reserved																																															
0011	/8	50 MHz																																															
0100	/10	40 MHz																																															
0101	/12	33.33 MHz																																															
0110	/14	28.57 MHz																																															
0111	/16	25 MHz																																															
1000	/18	22.22 MHz																																															
1001	/20	20 MHz																																															
1010	/22	18.18 MHz																																															
1011	/24	16.67 MHz																																															
1100	/26	15.38 MHz																																															
1101	/28	14.29 MHz																																															
1110	/30	13.33 MHz																																															
1111	/32	12.5 MHz (default)																																															
22	USESYSCLK	R/W	0	Use the system clock divider as the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source.																																													
21:14	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.																																													
13	PWRDN	R/W	1	<p>PLL Power Down</p> <p>This bit connects to the PLL PWRDN input. The reset value of 1 powers down the PLL.</p>																																													
12	reserved	RO	1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																																													

Bit/Field	Name	Type	Reset	Description										
11	BYPASS	R/W	1	<p>PLL Bypass</p> <p>Chooses whether the system clock is derived from the PLL output or the OSC source. If set, the clock that drives the system is the OSC source. Otherwise, the clock that drives the system is the PLL output clock divided by the system divider.</p> <p>Note: The ADC must be clocked from the PLL or directly from a 14-MHz to 18-MHz clock source to operate properly. While the ADC works in a 14-18 MHz range, to maintain a 1 M sample/second rate, the ADC must be provided a 16-MHz clock source.</p>										
10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
9:6	XTAL	R/W	0xB	This field specifies the crystal value attached to the main oscillator. The encoding for this field is provided in Table 6-3 on page 75.										
5:4	OSCSRC	R/W	0x1	<p>Picks among the four input sources for the OSC. The values are:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Input Source</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>Main oscillator (default)</td> </tr> <tr> <td>01</td> <td>Internal oscillator (default)</td> </tr> <tr> <td>10</td> <td>Internal oscillator / 4 (this is necessary if used as input to PLL)</td> </tr> <tr> <td>11</td> <td>reserved</td> </tr> </tbody> </table>	Value	Input Source	00	Main oscillator (default)	01	Internal oscillator (default)	10	Internal oscillator / 4 (this is necessary if used as input to PLL)	11	reserved
Value	Input Source													
00	Main oscillator (default)													
01	Internal oscillator (default)													
10	Internal oscillator / 4 (this is necessary if used as input to PLL)													
11	reserved													
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
1	IOSCDIS	R/W	0	<p>Internal Oscillator (IOSC) Disable</p> <p>0: Internal oscillator is enabled.</p> <p>1: Internal oscillator is disabled.</p>										
0	MOSCDIS	R/W	1	<p>Main Oscillator Disable</p> <p>0: Main oscillator is enabled.</p> <p>1: Main oscillator is disabled (default).</p>										

Table 6-3. Default Crystal Field Values and PLL Programming

Crystal Number (XTAL Binary Value)	Crystal Frequency (MHz) Not Using the PLL	Crystal Frequency (MHz) Using the PLL
0000	1.000	reserved
0001	1.8432	reserved
0010	2.000	reserved
0011	2.4576	reserved
0100	3.579545 MHz	
0101	3.6864 MHz	
0110	4 MHz	
0111	4.096 MHz	

Crystal Number (XTAL Binary Value)	Crystal Frequency (MHz) Not Using the PLL	Crystal Frequency (MHz) Using the PLL
1000		4.9152 MHz
1001		5 MHz
1010		5.12 MHz
1011		6 MHz (reset value)
1100		6.144 MHz
1101		7.3728 MHz
1110		8 MHz
1111		8.192 MHz

DRAFT

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

This register provides a means of translating external crystal frequencies into the appropriate PLL settings. This register is initialized during the reset sequence and updated anytime that the `XTAL` field changes in the **Run-Mode Clock Configuration (RCC)** register (see page 73).

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

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

XTAL to PLL Translation (PLLCFG)

Base 0x400F.E000

Offset 0x064

Type RO, reset -

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

Bit/Field	Name	Type	Reset	Description
31:16	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.
15:14	OD	RO	-	This field specifies the value supplied to the PLL's OD input.
13:5	F	RO	-	This field specifies the value supplied to the PLL's F input.
4:0	R	RO	-	This field specifies the value supplied to the PLL's R input.

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

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

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

Run-Mode Clock Configuration 2 (RCC2)

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

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

Bit/Field	Name	Type	Reset	Description
31	USERCC2	R/W	0	When set, overrides the RCC register fields.
30:29	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.
28:23	SYSDIV2	R/W	0x0F	System Clock Divisor (6-bit) Specifies which divisor is used to generate the system clock from the PLL output. The PLL VCO frequency is 400 MHz. This field is wider than the RCC register <code>SYSDIV</code> field in order to provide additional divisor values. This permits the system clock to be run at much lower frequencies during Deep Sleep mode. For example, where the RCC register <code>SYSDIV</code> encoding of 111 provides /16, the RCC2 register <code>SYSDIV2</code> encoding of 111111 provides /64.
22:14	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	PWRDN2	R/W	1	When set, powers down the PLL.
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	BYPASS2	R/W	1	When set, bypasses the PLL for the clock source.
10:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description																		
6:4	OSCSRC2	R/W	0	System Clock Source <table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>MOSC</td> <td>0</td> <td>Main oscillator</td> </tr> <tr> <td>IOSC</td> <td>1</td> <td>Internal oscillator</td> </tr> <tr> <td>IOSC/4</td> <td>2</td> <td>Internal oscillator / 4</td> </tr> <tr> <td>30kHz</td> <td>3</td> <td>30 kHz internal oscillator</td> </tr> <tr> <td>32kHz</td> <td>7</td> <td>32 kHz external oscillator</td> </tr> </tbody> </table>	Name	Value	Description	MOSC	0	Main oscillator	IOSC	1	Internal oscillator	IOSC/4	2	Internal oscillator / 4	30kHz	3	30 kHz internal oscillator	32kHz	7	32 kHz external oscillator
Name	Value	Description																				
MOSC	0	Main oscillator																				
IOSC	1	Internal oscillator																				
IOSC/4	2	Internal oscillator / 4																				
30kHz	3	30 kHz internal oscillator																				
32kHz	7	32 kHz external oscillator																				
3:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																		

DRAFT

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

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

Deep Sleep Clock Configuration (DSLPCCLKCFG)

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

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

Bit/Field	Name	Type	Reset	Description															
31:29	reserved	RO	0	Software should not rely on the value of 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	6-bit system divider field to override when Deep-Sleep occurs with PLL running.															
22:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.															
6:4	DSOSCSRC	R/W	0	When set, forces IOSC to be clock source during Deep Sleep mode.															
				<table border="1"> <thead> <tr> <th>Name</th> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>NOORIDE</td> <td>0</td> <td>No override to the oscillator clock source is done</td> </tr> <tr> <td>IOSC</td> <td>1</td> <td>Use internal 12 MHz oscillator as source</td> </tr> <tr> <td>30kHz</td> <td>3</td> <td>Use 30 kHz internal oscillator</td> </tr> <tr> <td>32kHz</td> <td>7</td> <td>Use 32 kHz external oscillator</td> </tr> </tbody> </table>	Name	Value	Description	NOORIDE	0	No override to the oscillator clock source is done	IOSC	1	Use internal 12 MHz oscillator as source	30kHz	3	Use 30 kHz internal oscillator	32kHz	7	Use 32 kHz external oscillator
Name	Value	Description																	
NOORIDE	0	No override to the oscillator clock source is done																	
IOSC	1	Use internal 12 MHz oscillator as source																	
30kHz	3	Use 30 kHz internal oscillator																	
32kHz	7	Use 32 kHz external oscillator																	
3:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.															

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

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

Device Identification 1 (DID1)

Base 0x400F.E000

Offset 0x004

Type RO, reset -

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

Bit/Field	Name	Type	Reset	Description				
31:28	VER	RO	0x1	<p>This field defines the DID1 register format version. The version number is numeric. The value of the <code>VER</code> field is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x1</td> <td>First revision of the DID1 register format, indicating a Stellaris LM3Snnnn device.</td> </tr> </tbody> </table>	Value	Description	0x1	First revision of the DID1 register format, indicating a Stellaris LM3Snnnn device.
Value	Description							
0x1	First revision of the DID1 register format, indicating a Stellaris LM3Snnnn device.							
27:24	FAM	RO	0x0	<p>Family</p> <p>This field provides the family identification of the device within the Luminary Micro product portfolio. The value is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Stellaris family of microcontrollers, that is, all devices with external part numbers starting with LM3S.</td> </tr> </tbody> </table>	Value	Description	0x0	Stellaris family of microcontrollers, that is, all devices with external part numbers starting with LM3S.
Value	Description							
0x0	Stellaris family of microcontrollers, that is, all devices with external part numbers starting with LM3S.							
23:16	PARTNO	RO	0x88	<p>Part Number</p> <p>This field provides the part number of the device within the family. The value is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x88</td> <td>LM3S8938</td> </tr> </tbody> </table>	Value	Description	0x88	LM3S8938
Value	Description							
0x88	LM3S8938							
15:13	PINCOUNT	RO	0x2	<p>Package Pin Count</p> <p>This field specifies the number of pins on the device package. The value is encoded as follows (all other encodings are reserved):</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x2</td> <td>100-pin package</td> </tr> </tbody> </table>	Value	Description	0x2	100-pin package
Value	Description							
0x2	100-pin package							

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

Register 13: Device Capabilities 0 (DC0), offset 0x008

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

Device Capabilities 0 (DC0)

Base 0x400F.E000

Offset 0x008

Type RO, reset 0x00FF.007F

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

Bit/Field	Name	Type	Reset	Description
31:16	SRAMSZ	RO	0x00FF	SRAM Size Indicates the size of the on-chip SRAM memory. Value Description 0x00FF 64 KB of SRAM
15:0	FLASHSZ	RO	0x007F	Flash Size Indicates the size of the on-chip flash memory. Value Description 0x007F 256 KB of Flash

Register 14: Device Capabilities 1 (DC1), offset 0x010

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

Device Capabilities 1 (DC1)

Base 0x400F.E000

Offset 0x010

Type RO, reset 0x0101.33FF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved							CAN0	reserved							SARADC0
Type	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	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SYSDIV				MAXADCSPD				MPU	HIB	TEMPSNS	PLL	WDT	SWO	SWD	JTAG
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:25	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	CAN0	RO	1	When set, indicates that CAN unit 0 is present.
23:17	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	SARADC0	RO	1	When set, indicates that general SAR ADC 0 is present.
15:12	SYSDIV	RO	0x3	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 <code>SYSDIV</code> bit.
				Value Description
				0x3 Specifies a 50-MHz CPU clock with a PLL divider of 4.
11:8	MAXADCSPD	RO	0x3	This field indicates the maximum rate at which the ADC samples data.
				Value Description
				0x3 1M samples/second
7	MPU	RO	1	When set, indicates that the Cortex-M3 Memory Protection Unit (MPU) module is present. See the ARM Cortex-M3 Technical Reference Manual for details on the MPU.
6	HIB	RO	1	When set, indicates that the Hibernation module is present.
5	TEMPSNS	RO	1	When set, indicates that the on-chip temperature sensor is present.
4	PLL	RO	1	When set, indicates that the on-chip Phase Locked Loop (PLL) is present.

Bit/Field	Name	Type	Reset	Description
3	WDT	RO	1	When set, indicates that a watchdog timer is present.
2	SWO	RO	1	When set, indicates that the Serial Wire Output (SWO) trace port is present.
1	SWD	RO	1	When set, indicates that the Serial Wire Debugger (SWD) is present.
0	JTAG	RO	1	When set, indicates that the JTAG debugger interface is present.

DRAFT

Register 15: Device Capabilities 2 (DC2), offset 0x014

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

Device Capabilities 2 (DC2)

Base 0x400F.E000

Offset 0x014

Type RO, reset 0x070F.5017

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

Bit/Field	Name	Type	Reset	Description
31:27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	COMP2	RO	1	When set, indicates that analog comparator 2 is present.
25	COMP1	RO	1	When set, indicates that analog comparator 1 is present.
24	COMP0	RO	1	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	When set, indicates that General-Purpose Timer module 3 is present.
18	TIMER2	RO	1	When set, indicates that General-Purpose Timer module 2 is present.
17	TIMER1	RO	1	When set, indicates that General-Purpose Timer module 1 is present.
16	TIMER0	RO	1	When set, indicates that General-Purpose Timer module 0 is present.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	RO	1	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	When set, indicates that I2C module 0 is present.
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	RO	1	When set, indicates that SSI module 0 is present.

Bit/Field	Name	Type	Reset	Description
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	When set, indicates that UART module 2 is present.
1	UART1	RO	1	When set, indicates that UART module 1 is present.
0	UART0	RO	1	When set, indicates that UART module 0 is present.

DRAFT

Register 16: Device Capabilities 3 (DC3), offset 0x018

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

Device Capabilities 3 (DC3)

Base 0x400F.E000

Offset 0x018

Type RO, reset 0x3FFF.7FC0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved		CCP5	CCP4	CCP3	CCP2	CCP1	CCP0	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	C2O	C2PLUS	C2MINUS	C1O	C1PLUS	C1MINUS	C0O	C0PLUS	C0MINUS	reserved					
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:30	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
29	CCP5	RO	1	When set, indicates that Capture/Compare/PWM pin 5 is present.
28	CCP4	RO	1	When set, indicates that Capture/Compare/PWM pin 4 is present.
27	CCP3	RO	1	When set, indicates that Capture/Compare/PWM pin 3 is present.
26	CCP2	RO	1	When set, indicates that Capture/Compare/PWM pin 2 is present.
25	CCP1	RO	1	When set, indicates that Capture/Compare/PWM pin 1 is present.
24	CCP0	RO	1	When set, indicates that Capture/Compare/PWM pin 0 is present.
23	ADC7	RO	1	When set, indicates that ADC pin 7 is present.
22	ADC6	RO	1	When set, indicates that ADC pin 6 is present.
21	ADC5	RO	1	When set, indicates that ADC pin 5 is present.
20	ADC4	RO	1	When set, indicates that ADC pin 4 is present.
19	ADC3	RO	1	When set, indicates that ADC pin 3 is present.
18	ADC2	RO	1	When set, indicates that ADC pin 2 is present.
17	ADC1	RO	1	When set, indicates that ADC pin 1 is present.
16	ADC0	RO	1	When set, indicates that ADC pin 0 is present.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	C2O	RO	1	When set, indicates that the analog comparator 2 output pin is present.
13	C2PLUS	RO	1	When set, indicates that the analog comparator 2 (+) input pin is present.
12	C2MINUS	RO	1	When set, indicates that the analog comparator 2 (-) input pin is present.

Bit/Field	Name	Type	Reset	Description
11	C1O	RO	1	When set, indicates that the analog comparator 1 output pin is present.
10	C1PLUS	RO	1	When set, indicates that the analog comparator 1 (+) input pin is present.
9	C1MINUS	RO	1	When set, indicates that the analog comparator 1 (-) input pin is present.
8	C0O	RO	1	When set, indicates that the analog comparator 0 output pin is present.
7	C0PLUS	RO	1	When set, indicates that the analog comparator 0 (+) input pin is present.
6	C0MINUS	RO	1	When set, indicates that the analog comparator 0 (-) input pin is present.
5:0	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.

DRAFT

Register 17: Device Capabilities 4 (DC4), offset 0x01C

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

Device Capabilities 4 (DC4)

Base 0x400F.E000
 Offset 0x01C
 Type RO, reset 0x5100.007F

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	EPHY0	reserved	EMAC0	reserved			E1588	reserved							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	1	0	1	0	0	0	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
	reserved									GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPHY0	RO	1	When set, indicates that Ethernet PHY module 0 is present.
29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	EMAC0	RO	1	When set, indicates that Ethernet MAC module 0 is present.
27:25	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	E1588	RO	1	When set, indicates that that EMAC0 is 1588-capable.
23:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	GPIOG	RO	1	When set, indicates that GPIO Port G is present.
5	GPIOF	RO	1	When set, indicates that GPIO Port F is present.
4	GPIOE	RO	1	When set, indicates that GPIO Port E is present.
3	GPIOD	RO	1	When set, indicates that GPIO Port D is present.
2	GPIOC	RO	1	When set, indicates that GPIO Port C is present.
1	GPIOB	RO	1	When set, indicates that GPIO Port B is present.
0	GPIOA	RO	1	When set, indicates that GPIO Port A is present.

Register 18: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100

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

Run Mode Clock Gating Control Register 0 (RCGC0)

Base 0x400F.E000

Offset 0x100

Type R/W, reset 0x00000040

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved							CAN0	reserved							SARADC0	
Type	RO	RO	RO	RO	RO	RO	RO	R/W	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved				MAXADCSPD			reserved	HIB	reserved	WDT	reserved					
Type	RO	RO	RO	RO	R/W	R/W	R/W	R/W	RO	R/W	RO	RO	R/W	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description
31:25	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	CAN0	R/W	0	This bit controls the clock gating for CAN unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled.
23:17	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	SARADC0	R/W	0	This bit controls the clock gating for SAR ADC module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, a read or write to the unit generates a bus fault.
15:12	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										
11:8	MAXADCSPD	R/W	0	<p>This field sets the rate at which the ADC samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADCSPD bit as follows:</p> <table border="1"> <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													
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	This bit controls the clock gating for the Hibernation module. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled.										
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
3	WDT	R/W	0	This bit controls the clock gating for the WDT module. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, a read or write to the unit generates a bus fault.										
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										

Register 19: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110

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

Sleep Mode Clock Gating Control Register 0 (SCGC0)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved							CAN0	reserved							SARADC0	
Type	RO	RO	RO	RO	RO	RO	RO	R/W	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved				MAXADCSPD			reserved	HIB	reserved	WDT	reserved					
Type	RO	RO	RO	RO	R/W	R/W	R/W	R/W	RO	R/W	RO	RO	R/W	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description
31:25	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	CAN0	R/W	0	This bit controls the clock gating for CAN unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled.
23:17	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	SARADC0	R/W	0	This bit controls the clock gating for general SAR ADC module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, a read or write to the unit generates a bus fault.
15:12	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										
11:8	MAXADCSPD	R/W	0	<p>This field sets the rate at which the ADC samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADCSPD bit as follows:</p> <table border="1"> <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													
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	This bit controls the clock gating for the Hibernation module. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled.										
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
3	WDT	R/W	0	This bit controls the clock gating for the WDT module. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, a read or write to the unit generates a bus fault.										
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										

Register 20: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120

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

Deep Sleep Mode Clock Gating Control Register 0 (DCGC0)

Base 0x400F.E000

Offset 0x120

Type R/W, reset 0x00000040

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved							CAN0	reserved							SARADC0	
Type	RO	RO	RO	RO	RO	RO	RO	R/W	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	reserved				MAXADCSPD			reserved	HIB	reserved	WDT	reserved					
Type	RO	RO	RO	RO	R/W	R/W	R/W	R/W	RO	R/W	RO	RO	R/W	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Bit/Field	Name	Type	Reset	Description
31:25	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	CAN0	R/W	0	This bit controls the clock gating for CAN unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled.
23:17	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	SARADC0	R/W	0	This bit controls the clock gating for general SAR ADC module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, a read or write to the unit generates a bus fault.
15:12	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										
11:8	MAXADCSPD	R/W	0	<p>This field sets the rate at which the ADC samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADCSPD bit as follows:</p> <table border="1"> <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													
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	This bit controls the clock gating for the Hibernation module. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled.										
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
3	WDT	R/W	0	This bit controls the clock gating for the WDT module. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, a read or write to the unit generates a bus fault.										
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										

Register 21: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104

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

Run Mode Clock Gating Control Register 1 (RCGC1)

Base 0x400F.E000

Offset 0x104

Type R/W, reset 0x00000000

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

Bit/Field	Name	Type	Reset	Description
31:27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	COMP2	R/W	0	This bit controls the clock gating for analog comparator 2. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
25	COMP1	R/W	0	This bit controls the clock gating for analog comparator 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
24	COMP0	R/W	0	This bit controls the clock gating for analog comparator 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
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	This bit controls the clock gating for General-Purpose Timer module 3. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

Bit/Field	Name	Type	Reset	Description
18	TIMER2	R/W	0	This bit controls the clock gating for General-Purpose Timer module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
17	TIMER1	R/W	0	This bit controls the clock gating for General-Purpose Timer module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
16	TIMER0	R/W	0	This bit controls the clock gating for General-Purpose Timer module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	This bit controls the clock gating for I2C module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	This bit controls the clock gating for I2C module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	R/W	0	This bit controls the clock gating for SSI module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	R/W	0	This bit controls the clock gating for UART module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
1	UART1	R/W	0	This bit controls the clock gating for UART module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
0	UART0	R/W	0	This bit controls the clock gating for UART module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

Register 22: Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114

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

Sleep Mode Clock Gating Control Register 1 (SCGC1)

Base 0x400F.E000

Offset 0x114

Type R/W, reset 0x00000000

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

Bit/Field	Name	Type	Reset	Description
31:27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	COMP2	R/W	0	This bit controls the clock gating for analog comparator 2. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
25	COMP1	R/W	0	This bit controls the clock gating for analog comparator 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
24	COMP0	R/W	0	This bit controls the clock gating for analog comparator 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
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	This bit controls the clock gating for General-Purpose Timer module 3. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

Bit/Field	Name	Type	Reset	Description
18	TIMER2	R/W	0	This bit controls the clock gating for General-Purpose Timer module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
17	TIMER1	R/W	0	This bit controls the clock gating for General-Purpose Timer module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
16	TIMER0	R/W	0	This bit controls the clock gating for General-Purpose Timer module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	This bit controls the clock gating for I2C module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	This bit controls the clock gating for I2C module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	R/W	0	This bit controls the clock gating for SSI module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	R/W	0	This bit controls the clock gating for UART module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
1	UART1	R/W	0	This bit controls the clock gating for UART module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
0	UART0	R/W	0	This bit controls the clock gating for UART module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

Register 23: Deep Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124

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

Deep Sleep Mode Clock Gating Control Register 1 (DCGC1)

Base 0x400F.E000

Offset 0x124

Type R/W, reset 0x00000000

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

Bit/Field	Name	Type	Reset	Description
31:27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	COMP2	R/W	0	This bit controls the clock gating for analog comparator 2. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
25	COMP1	R/W	0	This bit controls the clock gating for analog comparator 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
24	COMP0	R/W	0	This bit controls the clock gating for analog comparator 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
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	This bit controls the clock gating for General-Purpose Timer module 3. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

Bit/Field	Name	Type	Reset	Description
18	TIMER2	R/W	0	This bit controls the clock gating for General-Purpose Timer module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
17	TIMER1	R/W	0	This bit controls the clock gating for General-Purpose Timer module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
16	TIMER0	R/W	0	This bit controls the clock gating for General-Purpose Timer module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	This bit controls the clock gating for I2C module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	This bit controls the clock gating for I2C module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	R/W	0	This bit controls the clock gating for SSI module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	R/W	0	This bit controls the clock gating for UART module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
1	UART1	R/W	0	This bit controls the clock gating for UART module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
0	UART0	R/W	0	This bit controls the clock gating for UART module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

Register 24: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108

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

Run Mode Clock Gating Control Register 2 (RCGC2)

Base 0x400F.E000

Offset 0x108

Type R/W, reset 0x00000000

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

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPHY0	R/W	0	This bit controls the clock gating for Ethernet PHY unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
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	This bit controls the clock gating for Ethernet MAC unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
27:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	GPIOG	R/W	0	This bit controls the clock gating for Port G. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
5	GPIOF	R/W	0	This bit controls the clock gating for Port F. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

Bit/Field	Name	Type	Reset	Description
4	GPIOE	R/W	0	This bit controls the clock gating for Port E. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
3	GPIOD	R/W	0	This bit controls the clock gating for Port D. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
2	GPIOC	R/W	0	This bit controls the clock gating for Port C. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
1	GPIOB	R/W	0	This bit controls the clock gating for Port B. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
0	GPIOA	R/W	0	This bit controls the clock gating for Port A. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

DRAFT

Register 25: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118

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

Sleep Mode Clock Gating Control Register 2 (SCGC2)

Base 0x400F.E000

Offset 0x118

Type R/W, reset 0x00000000

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

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of 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	This bit controls the clock gating for Ethernet PHY unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
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	This bit controls the clock gating for Ethernet MAC unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
27:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	GPIOG	R/W	0	This bit controls the clock gating for Port G. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

Bit/Field	Name	Type	Reset	Description
5	GPIOF	R/W	0	This bit controls the clock gating for Port F. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
4	GPIOE	R/W	0	This bit controls the clock gating for Port E. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
3	GPIOD	R/W	0	This bit controls the clock gating for Port D. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
2	GPIOC	R/W	0	This bit controls the clock gating for Port C. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
1	GPIOB	R/W	0	This bit controls the clock gating for Port B. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
0	GPIOA	R/W	0	This bit controls the clock gating for Port A. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

Register 26: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128

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

Deep Sleep Mode Clock Gating Control Register 2 (DCGC2)

Base 0x400F.E000

Offset 0x128

Type R/W, reset 0x00000000

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

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of 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	This bit controls the clock gating for Ethernet PHY unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
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	This bit controls the clock gating for Ethernet MAC unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
27:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	GPIOG	R/W	0	This bit controls the clock gating for Port G. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

Bit/Field	Name	Type	Reset	Description
5	GPIOF	R/W	0	This bit controls the clock gating for Port F. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
4	GPIOE	R/W	0	This bit controls the clock gating for Port E. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
3	GPIOD	R/W	0	This bit controls the clock gating for Port D. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
2	GPIOC	R/W	0	This bit controls the clock gating for Port C. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
1	GPIOB	R/W	0	This bit controls the clock gating for Port B. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.
0	GPIOA	R/W	0	This bit controls the clock gating for Port A. If set, the unit receives a clock and functions. Otherwise, the unit is unlocked and disabled. If the unit is unlocked, reads or writes to the unit will generate a bus fault.

Register 27: Software Reset Control 0 (SRCR0), offset 0x040Writes to this register are masked by the bits in the **Device Capabilities 1 (DC1)** register.

Software Reset Control 0 (SRCR0)

Base 0x400F.E000

Offset 0x040

Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved							CAN0	reserved							SARADC0
Type	RO	RO	RO	RO	RO	RO	RO	R/W	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved							HIB	reserved			WDT	reserved			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:25	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	CAN0	R/W	0	Reset control for CAN unit 0.
23:17	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	SARADC0	R/W	0	Reset control for SAR ADC module 0.
15:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	HIB	R/W	0	Reset control for the Hibernation module.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT	R/W	0	Reset control for Watchdog unit.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 28: Software Reset Control 1 (SRCR1), offset 0x044Writes to this register are masked by the bits in the **Device Capabilities 2 (DC2)** register.

Software Reset Control 1 (SRCR1)

Base 0x400F.E000

Offset 0x044

Type R/W, reset 0x00000000

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

Bit/Field	Name	Type	Reset	Description
31:27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	COMP2	R/W	0	Reset control for analog comparator 2.
25	COMP1	R/W	0	Reset control for analog comparator 1.
24	COMP0	R/W	0	Reset control for analog comparator 0.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Reset control for General-Purpose Timer module 3.
18	TIMER2	R/W	0	Reset control for General-Purpose Timer module 2.
17	TIMER1	R/W	0	Reset control for General-Purpose Timer module 1.
16	TIMER0	R/W	0	Reset control for General-Purpose Timer module 0.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	Reset control for I2C unit 1.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	Reset control for I2C unit 0.
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	R/W	0	Reset control for SSI unit 0.

Bit/Field	Name	Type	Reset	Description
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	Reset control for UART unit 2.
1	UART1	R/W	0	Reset control for UART unit 1.
0	UART0	R/W	0	Reset control for UART unit 0.

DRAFT

Register 29: Software Reset Control 2 (SRCR2), offset 0x048

Writes to this register are masked by the bits in the **Device Capabilities 4 (DC4)** register.

Software Reset Control 2 (SRCR2)

Base 0x400F.E000
 Offset 0x048
 Type R/W, reset 0x00000000

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

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30	EPHY0	R/W	0	Reset control for Ethernet PHY unit 0.
29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	EMAC0	R/W	0	Reset control for Ethernet MAC unit 0.
27:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	GPIOG	R/W	0	Reset control for GPIO Port G.
5	GPIOF	R/W	0	Reset control for GPIO Port F.
4	GPIOE	R/W	0	Reset control for GPIO Port E.
3	GPIOD	R/W	0	Reset control for GPIO Port D.
2	GPIOC	R/W	0	Reset control for GPIO Port C.
1	GPIOB	R/W	0	Reset control for GPIO Port B.
0	GPIOA	R/W	0	Reset control for GPIO Port A.

7 Hibernation Module

HIB

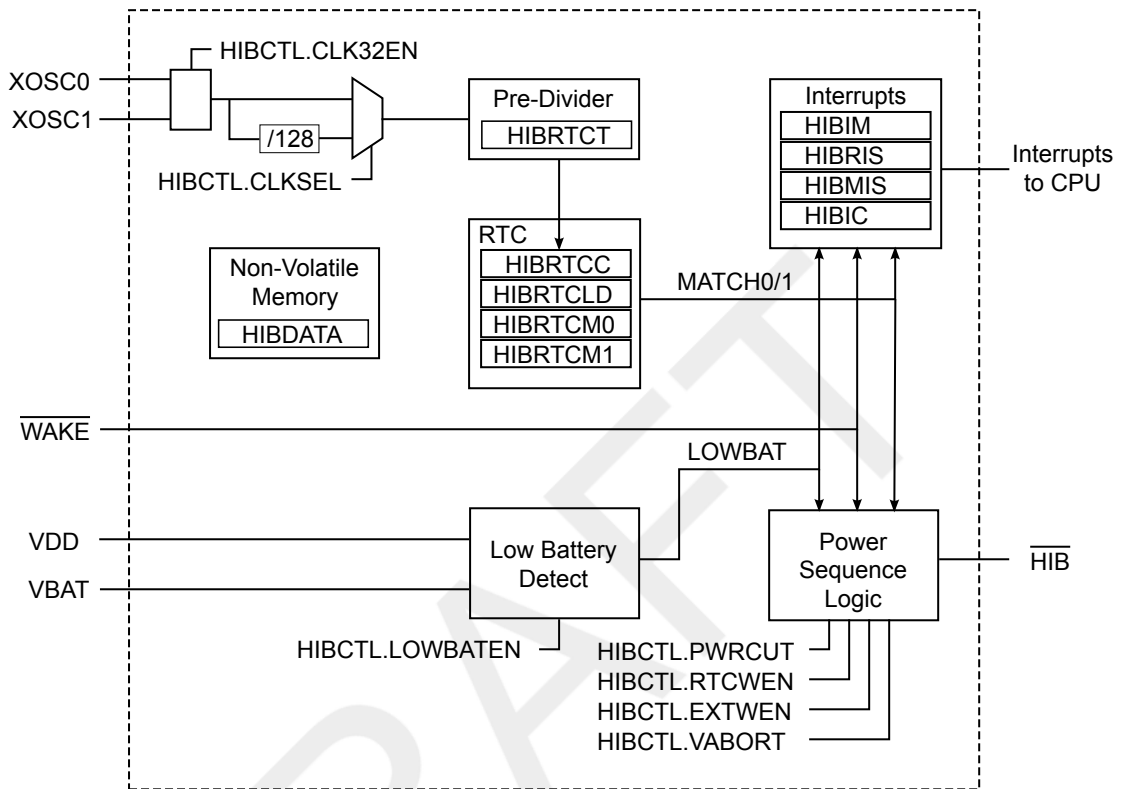
The Hibernation Module manages removal and restoration of power to the rest of the microcontroller to provide a means for reducing power consumption. When the processor and peripherals are idle, power can be completely removed with only the Hibernation Module remaining powered. Power can be restored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

The Hibernation module has the following features:

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

7.1 Block Diagram

Figure 7-1. Hibernation Module Block Diagram



7.2 Functional Description

The Hibernation module controls the power to the processor with an enable signal (\overline{HIB}) that signals an external voltage regulator to turn off. The Hibernation module itself is powered from a separate supply such as a battery or auxiliary supply. It also has a separate clock source to maintain a real-time clock (RTC). Once in hibernation, the module signals an external voltage regulator to turn back on the power when an external pin (\overline{WAKE}) is asserted, or when the internal RTC reaches a certain value. The Hibernation module can also detect when the battery voltage is low, and optionally prevent hibernation when this occurs.

Power-up from a power cut to code execution is defined as the regulator turn-on time (specified at 250 μ s maximum) plus the normal chip POR (see Figure 22-12 on page 524).

7.2.1 Register Access Timing

Because the Hibernation module has an independent clocking domain, certain registers must be written only with a timing gap between accesses. The delay time is $t_{HIB_REG_WRITE}$, therefore software must guarantee that a delay of $t_{HIB_REG_WRITE}$ is inserted between back-to-back writes to certain Hibernation registers, or between a write followed by a read to those same registers. There is no restriction on timing for back-to-back reads from the Hibernation module. Refer to "Register Descriptions" on page 118 for details about which registers are subject to this timing restriction.

7.2.2 Clock Source

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

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

7.2.3 Battery Management

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

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

The Hibernation module can be configured to detect a low battery condition by setting the LOWBATEN bit of the HIBCTL register. In this configuration, the LOWBAT bit of the HIBRIS register will be set when the battery level is low. If the VABORT bit is also set, then the module is prevented from entering Hibernation mode when a low battery is detected. The module can also be configured to generate an interrupt for the low-battery condition (see “Interrupts and Status” on page 116).

7.2.4 Real-Time Clock

The Hibernation module includes a 32-bit counter that increments once per second with a proper clock source and configuration (see “Clock Source” on page 115). The 32.768-kHz clock signal is fed into a trim predivider which counts down from a nominal value of 0x7FFF to achieve a once per second clock rate for the RTC. The trim predivider register can be adjusted up or down to compensate for inaccuracies in the clock source. The trim predivider 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 trim predivider can be adjusted by reading and writing the HIBRTCT register. The predivider is updated once every 64 seconds from this register. 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 116).

7.2.5 Non-Volatile Memory

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

7.2.6 Power Control

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

The Hibernation module is configured to wake from the external $\overline{\text{WAKE}}$ pin by setting the **PINWEN** bit of the **HIBCTL** register. It is configured to wake from RTC match by setting the **RTCWEN** bit. Either one or both of these bits can be set prior to going into hibernation.

When the Hibernation module wakes, the microcontroller will see a normal power-on reset. It can detect that the power-on was due to a wake from hibernation by examining the raw interrupt status register (see “Interrupts and Status” on page 116) and by looking for state data in the non-volatile memory (see “Non-Volatile Memory” on page 116).

7.2.7 Interrupts and Status

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

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

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

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

7.3 Initialization and Configuration

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

Timing” on page 114). The registers that require a delay are denoted with a footnote in Table 7-1 on page 118.

7.3.1 Initialization

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

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

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

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

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

7.3.2 RTC Match Functionality (No Hibernation)

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

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

7.3.3 RTC Match/Wake-Up from Hibernation

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

1. Write the required RTC match value to the **RTCMn** 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 0x030-0x130.
4. Set the RTC Match Wake-Up and start the hibernation sequence by writing 0x0000.004F to the **HIBCTL** register at offset 0x010.

7.3.4 External Wake-Up from Hibernation

The following steps are needed to use the Hibernation module with the external \overline{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 0x030-0x130.
2. Enable the external wake and start the hibernation sequence by writing 0x0000.0056 to the **HIBCTL** register at offset 0x010.

7.3.5 RTC/External Wake-Up from Hibernation

1. Write the required RTC match value to the **RTCMn** 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 0x030-0x130.
4. Set the RTC Match/External Wake-Up and start the hibernation sequence by writing 0x0000.005F to the **HIBCTL** register at offset 0x010.

7.4 Register Map

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are internal BAPI module registers on the VBAPI voltage domain and the 32-kHz clock domain.

Table 7-1. Hibernation Module Register Map

Offset	Name	Type	Reset	Description	See page
0x000	HIBRTCC	RO	0x0000.0000	Hibernation RTC Counter	119
0x004	HIBRTCM0	R/W	0xFFFF.FFFF	Hibernation RTC Match 0	120
0x008	HIBRTCM1	R/W	0xFFFF.FFFF	Hibernation RTC Match 1	121
0x00C	HIBRTCLD	R/W	0xFFFF.FFFF	Hibernation RTC Load	122
0x010	HIBCTL	R/W	0x0000.0000	Hibernation Control	123
0x014	HIBIM	R/W	0x0000.0000	Hibernation Interrupt Mask	125
0x018	HIBRIS	RO	0x0000.0000	Hibernation Raw Interrupt Status	126
0x01C	HIBMIS	RO	0x0000.0000	Hibernation Masked Interrupt Status	127
0x020	HIBIC	W1C	0x0000.0000	Hibernation Interrupt Clear	128
0x024	HIBRTCT	R/W	0x0000.0000	Hibernation RTC Trim	129
0x030-0x12C	HIBDATA	R/W	0x0000.0000	Hibernation Data	130

7.5 Register Descriptions

All addresses given are relative to the Hibernation module Base Address at 0x400F.C000.

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

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

Hibernation RTC Counter (HIBRTCC)

Offset 0x000

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RTCC															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RTCC															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	RTCC	RO	0x0000.0000	RTC Counter

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

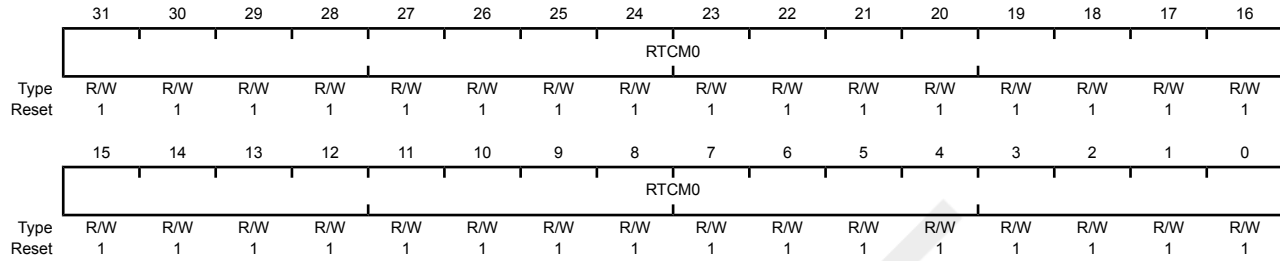
Register 2: Hibernation RTC Match 0 (HIBRTCM0), offset 0x004

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

Hibernation RTC Match 0 (HIBRTCM0)

Offset 0x004

Type R/W, reset 0xFFFF.FFFF



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

A write loads the value into the RTC match register.
A read returns the current match value.

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

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

Hibernation RTC Match 1 (HIBRTCM1)

Offset 0x008

Type R/W, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RTCM1															
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
	RTCM1															
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	RTCM1	R/W	0xFFFF.FFFF	RTC Match 1

A write loads the value into the RTC match register.

A read returns the current match value.

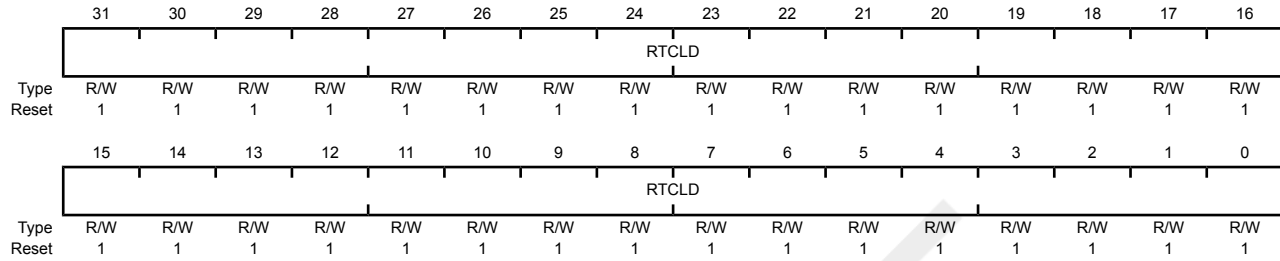
Register 4: Hibernation RTC Load (HIBRTCLD), offset 0x00C

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

Hibernation RTC Load (HIBRTCLD)

Offset 0x00C

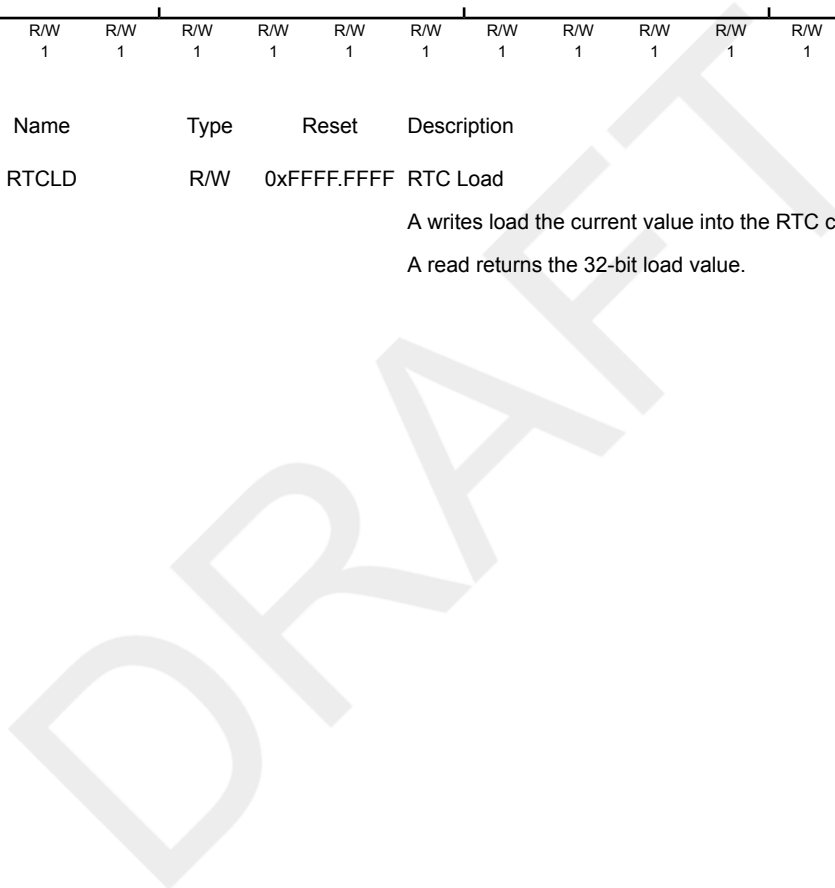
Type R/W, reset 0xFFFF.FFFF



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

A writes load the current value into the RTC counter (RTCC).

A read returns the 32-bit load value.



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

This register is the control register for the Hibernation module.

Hibernation Control (HIBCTL)

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								VABORT	CLK32EN	LOWBATEN	PINWEN	RTCWEN	CLKSEL	HIBREQ	RTCEN
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	VABORT	R/W	0	Power Cut Abort Enable 0: Power Cut occurs during a low-battery alert 1: Power Cut is aborted
6	CLK32EN	R/W	0	32-kHz Oscillator Enable 0: Disabled 1: Enabled This bit must be enabled to use the Hibernation module. If a crystal is used, then software should wait 20 ms after setting this bit to allow the crystal to power up and stabilize.
5	LOWBATEN	R/W	0	LOW BAT Monitoring Enable 0: Disabled 1: Enabled When set, low battery voltage detection is enabled.
4	PINWEN	R/W	0	External $\overline{\text{WAKE}}$ Pin Enable 0: Disabled 1: Enabled When set, an external event on the $\overline{\text{WAKE}}$ pin will re-power the device.
3	RTCWEN	R/W	0	RTC Wake-up Enable 0: Disabled 1: Enabled When set, an RTC match event (RTC0 or RTC1) will re-power the device based on the RTC counter value matching the corresponding match register 0 or 1.

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

DRAFT

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

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

Hibernation Interrupt Mask (HIBIM)

Offset 0x014

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												EXTW	LOWBAT	RTCALT1	RTCALT0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	R/W	0	External Wake-Up Interrupt Mask 0: Masked 1: Unmasked
2	LOWBAT	R/W	0	Low Battery Voltage Interrupt Mask 0: Masked 1: Unmasked
1	RTCALT1	R/W	0	RTC Alert1 Interrupt Mask 0: Masked 1: Unmasked
0	RTCALT0	R/W	0	RTC Alert0 Interrupt Mask 0: Masked 1: Unmasked

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

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

Hibernation Raw Interrupt Status (HIBRIS)

Offset 0x018

Type RO, reset 0x0000.0000

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

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

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

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

Hibernation Masked Interrupt Status (HIBMIS)

Offset 0x01C

Type RO, reset 0x0000.0000

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

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

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

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

Hibernation Interrupt Clear (HIBIC)

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												EXTW	LOWBAT	RTCALT1	RTCALT0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

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

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

Hibernation RTC Trim (HIBRTCT)

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
	TRIM															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

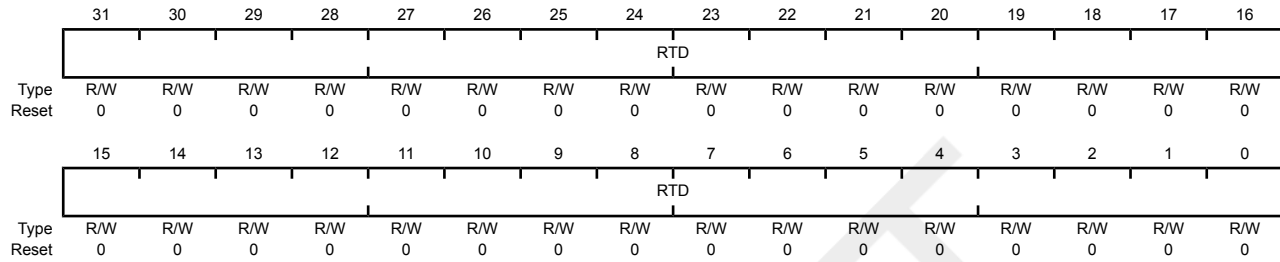
Register 11: Hibernation Data (HIBDATA), offset 0x030-0x12C

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

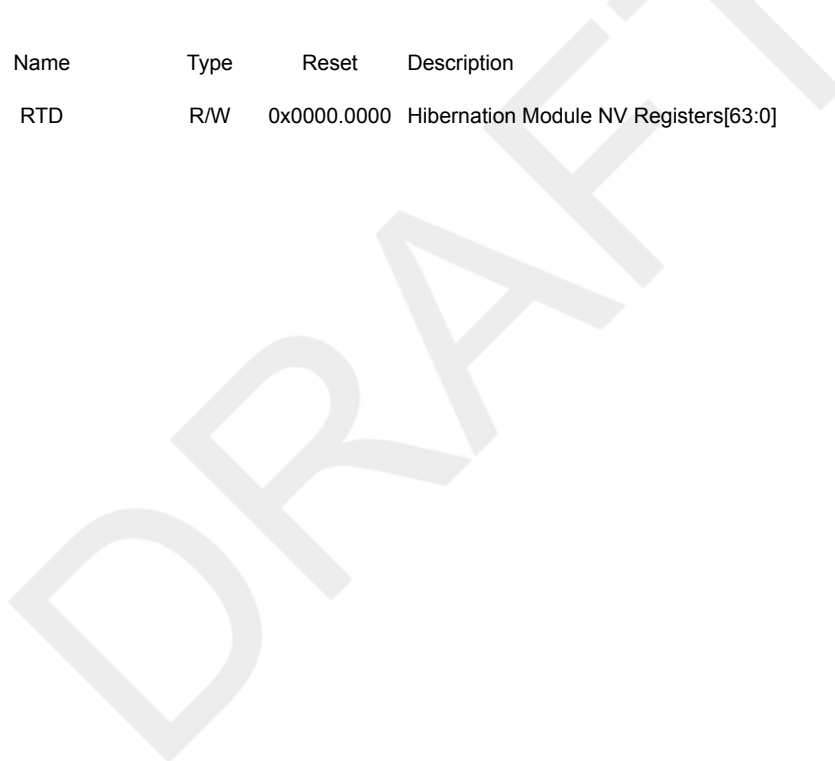
Hibernation Data (HIBDATA)

Offset 0x030-0x12C

Type R/W, reset 0x0000.0000



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



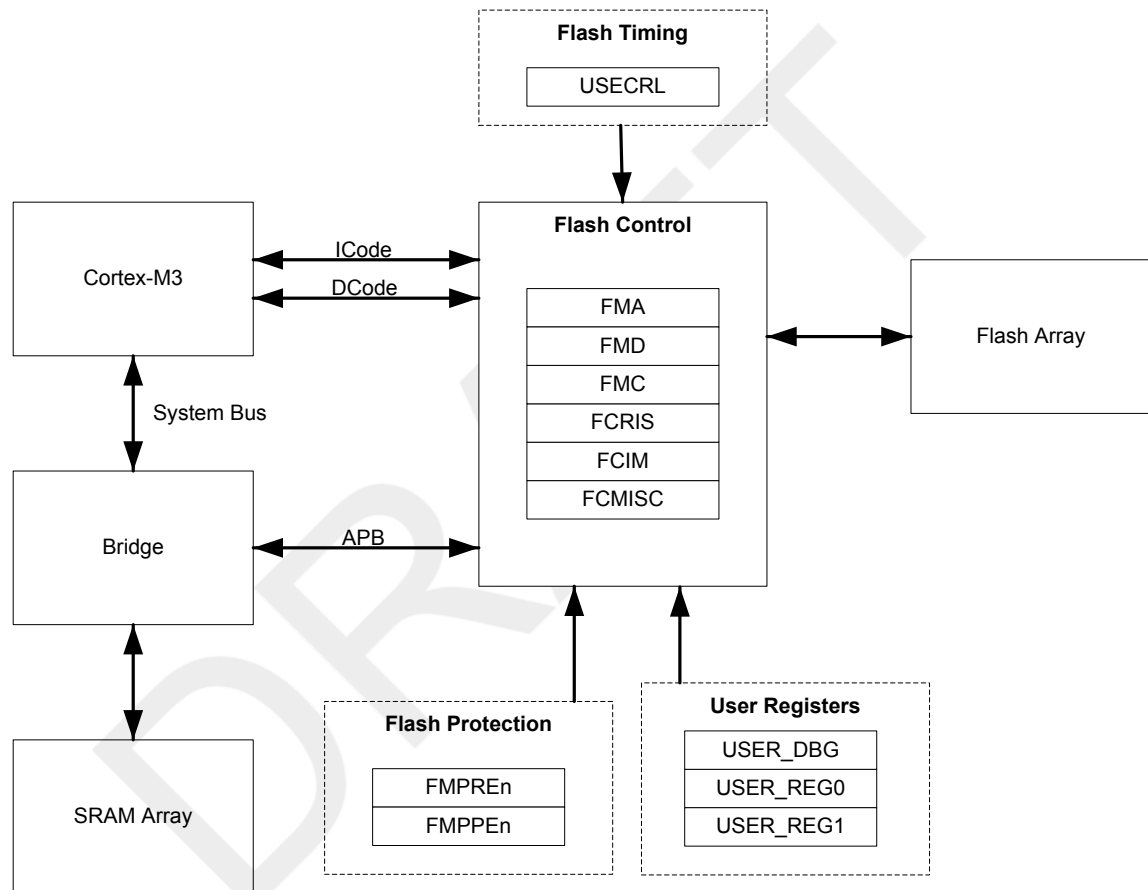
8 Internal Memory

FLASH

The LM3S8938 microcontroller comes with 64 KB of bit-banded SRAM and 256 KB of flash memory. The flash controller provides a user-friendly interface, making flash programming a simple task. Flash protection can be applied to the flash memory on a 2-KB block basis.

8.1 Block Diagram

Figure 8-1. Flash Block Diagram



8.2 Functional Description

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

8.2.1 SRAM Memory

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

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

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

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

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

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

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

8.2.2 Flash Memory

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

8.2.2.1 Flash Memory Timing

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

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

8.2.2.2 Flash Memory Protection

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

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

The policies may be combined as shown in Table 8-1 on page 133.

Table 8-1. Flash Protection Policy Combinations

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

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

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

The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This implements a policy of open access and programmability. The register bits may be changed by writing the specific register bit. The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. Details on programming these bits are discussed in “Nonvolatile Register Programming” on page 134.

8.3 Flash Memory Initialization and Configuration

8.3.1 Flash Programming

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

8.3.1.1 To program a 32-bit word:

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

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

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

8.3.1.3 To perform a mass erase of the flash:

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

8.3.2 Nonvolatile Register Programming

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

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

In addition, the **USER_REG0**, **USER_REG1**, and **USER_DBG** use bit 31 (**NOTWRITTEN**) of their respective registers to indicate that they are available for user write. These three registers can only be written once whereas the flash protection registers may be written multiple times. Table 8-2 on page 134 provides the FMA address required for commitment of each of the registers and the source of the data to be written when the **COMT** bit of the **FMC** register is written with a value of 0xA442.0008. After writing the **COMT** bit, the user may poll the **FMC** register to wait for the commit operation to complete.

Table 8-2. Flash Resident Registers^a

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

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

8.4 Register Map

Table 8-3 on page 134 lists the Flash memory and control registers. The offset listed is a hexadecimal increment to the register's address. The **FMA**, **FMD**, **FMC**, **FCRIS**, **FCIM**, and **FCMISC** registers are relative to the Flash control base address of 0x400F.D000. The **FMPREn**, **FMPPEn**, **USECRL**, **USER_DBG**, and **USER_REGn** registers are relative to the System Control base address of 0x400F.E000.

Note: A BV in the Reset column indicates the reset is a Build Value and part-specific. See the page number referenced for the reset value description.

Table 8-3. Internal Memory Register Map

Offset	Name	Type	Reset	Description	See page
Flash Control Offset					

Offset	Name	Type	Reset	Description	See page
0x000	FMA	R/W	0x0000.0000	Flash Memory Address	136
0x004	FMD	R/W	0x0000.0000	Flash Memory Data	137
0x008	FMC	R/W	0x0000.0000	Flash Memory Control	138
0x00C	FCRIS	RO	0x0000.0000	Flash Controller Raw Interrupt Status	140
0x010	FCIM	R/W	0x0000.0000	Flash Controller Interrupt Mask	141
0x014	FCMISC	R/W1C	0x0000.0000	Flash Controller Masked Interrupt Status and Clear	142
System Control Offset					
0x130	FMPRE0	R/W	BV	Flash Memory Protection Read Enable 0	144
0x200	FMPRE0	R/W	BV	Flash Memory Protection Read Enable 0	144
0x134	FMPPE0	R/W	BV	Flash Memory Protection Program Enable 0	145
0x400	FMPPE0	R/W	BV	Flash Memory Protection Program Enable 0	145
0x140	USECRL	R/W	0x31	USec Reload	143
0x1D0	USER_DBG	R/W	0xFFFF.FFFE	User Debug	146
0x1E0	USER_REG0	R/W	0x8FFF.FFFF	User Register 0	147
0x1E4	USER_REG1	R/W	0x8FFF.FFFF	User Register 1	148
0x204	FMPRE1	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 1	149
0x208	FMPRE2	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 2	150
0x20C	FMPRE3	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 3	151
0x404	FMPPE1	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 1	152
0x408	FMPPE2	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 2	153
0x40C	FMPPE3	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 3	154

8.5 Flash Register Descriptions (Flash Control Offset)

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

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

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

Flash Memory Address (FMA)

Base 0x400F.D000

Offset 0x000

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	OFFSET															
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
	OFFSET															
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	OFFSET	R/W	0x0	Address offset in flash where operation is performed, except for nonvolatile registers (see "Nonvolatile Register Programming" on page 134 for details on values for this field).

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

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

Flash Memory Data (FMD)

Base 0x400F.D000

Offset 0x004

Type R/W, reset 0x0000.0000

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

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

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

When this register is written, the flash controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 136). If the access is a write access, the data contained in the **Flash Memory Data (FMD)** register (see page 137) is written.

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

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

Flash Memory Control (FMC)

Base 0x400F.D000

Offset 0x008

Type R/W, reset 0x0000.0000

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

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

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

Register 4: Flash Controller Raw Interrupt Status (FCRIS), offset 0x00C

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

Flash Controller Raw Interrupt Status (FCRIS)

Base 0x400F.D000

Offset 0x00C

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PRIS	RO	0	<p>Programming Raw Interrupt Status</p> <p>This bit indicates the current state of the programming cycle. If set, the programming cycle completed; if cleared, the programming cycle has not completed. Programming cycles are either write or erase actions generated through the Flash Memory Control (FMC) register bits (see page 138).</p>
0	ARIS	RO	0	<p>Access Raw Interrupt Status</p> <p>This bit indicates if the flash was improperly accessed. If set, the program tried to access the flash counter to the policy as set in the Flash Memory Protection Read Enable (FMPREn) and Flash Memory Protection Program Enable (FMPPEn) registers. Otherwise, no access has tried to improperly access the flash.</p>

Register 5: Flash Controller Interrupt Mask (FCIM), offset 0x010

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

Flash Controller Interrupt Mask (FCIM)

Base 0x400F.D000

Offset 0x010

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														PMASK	AMASK
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PMASK	R/W	0	Programming Interrupt Mask This bit controls the reporting of the programming raw interrupt status to the controller. If set, a programming-generated interrupt is promoted to the controller. Otherwise, interrupts are recorded but suppressed from the controller.
0	AMASK	R/W	0	Access Interrupt Mask This bit controls the reporting of the access raw interrupt status to the controller. If set, an access-generated interrupt is promoted to the controller. Otherwise, interrupts are recorded but suppressed from the controller.

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

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

Flash Controller Masked Interrupt Status and Clear (FCMISC)

Base 0x400F.D000

Offset 0x014

Type R/W1C, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PMISC	R/W1C	0	Programming Masked Interrupt Status and Clear This bit indicates whether an interrupt was signaled because a programming cycle completed and was not masked. This bit is cleared by writing a 1. The <code>PRIS</code> bit in the <code>FCRIS</code> register (see page 140) is also cleared when the <code>PMISC</code> bit is cleared.
0	AMISC	R/W1C	0	Access Masked Interrupt Status and Clear This bit indicates whether an interrupt was signaled because an improper access was attempted and was not masked. This bit is cleared by writing a 1. The <code>ARIS</code> bit in the <code>FCRIS</code> register is also cleared when the <code>AMISC</code> bit is cleared.

8.6 Flash Register Descriptions (System Control Offset)

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

Register 7: USec Reload (USECRL), offset 0x140

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

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

USec Reload (USECRL)

Base 0x400F.E000

Offset 0x140

Type R/W, reset 0x31

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

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

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

Note: This register is aliased for backwards compatibility.

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

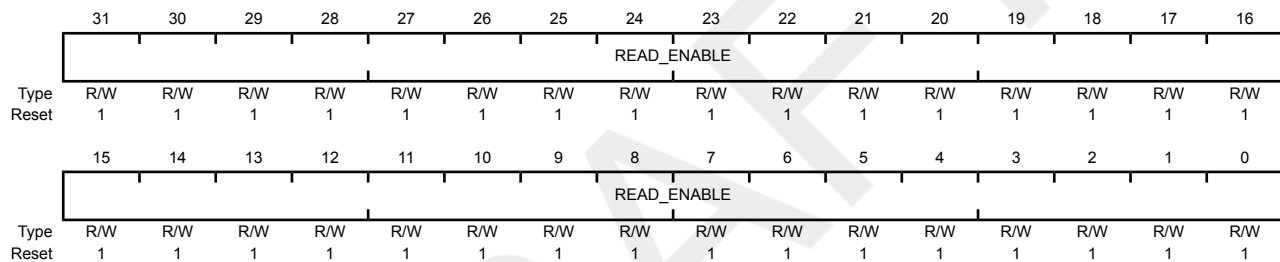
This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPREN** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves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

Flash Memory Protection Read Enable 0 (FMPRE0)

Base 0x400F.D000

Offset 0x130 and 0x200

Type R/W, reset 0xFFFF.FFFF



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

Value	Description
0xFFFFFFFF	Enables 256 KB of flash.

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

Note: This register is aliased for backwards compatibility.

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

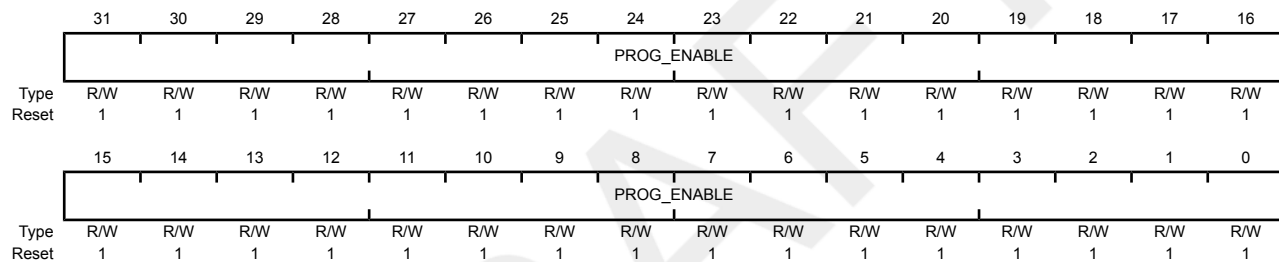
This register stores the execute-only protection bits for each 2-KB flash block (**FMPREN** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPREN** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves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

Flash Memory Protection Program Enable 0 (FMPPE0)

Base 0x400F.D000

Offset 0x134 and 0x400

Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Enables 2-KB flash blocks to be written or erased. The policies may be combined as shown in the table "Flash Protection Policy Combinations".

Value	Description
0xFFFFFFFF	Enables 256 KB of flash.

Register 10: User Debug (USER_DBG), offset 0x1D0

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

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

User Debug (USER_DBG)

Base 0x400F.E000

Offset 0x1D0

Type R/W, reset 0xFFFF.FFFE

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
	NOTWRITTEN	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															INIT1	DBG1	DBG0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0		

Bit/Field	Name	Type	Reset	Description
31	NOTWRITTEN	R/W	1	Specifies that this 32-bit dword has not been written.
30:3	DATA	R/W	0xFFFFFFFF	Contains the user data value. This field is initialized to all 1s and can only be written once.
2	INIT1	R/W	1	User data initialized to 1.
1	DBG1	R/W	1	The <code>DBG1</code> bit must be 1 and <code>DBG0</code> must be 0 for debug to be available.
0	DBG0	R/W	0	The <code>DBG1</code> bit must be 1 and <code>DBG0</code> must be 0 for debug to be available.

Register 11: User Register 0 (USER_REG0), offset 0x1E0

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

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

User Register 0 (USER_REG0)

Base 0x400F.E000

Offset 0x1E0

Type R/W, reset 0x8FFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NOTWRITTEN	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	0	0	0	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	NOTWRITTEN	R/W	1	Specifies that this 32-bit dword has not been written.
30:0	DATA	R/W	0xFFFFFFFF	Contains the user data value. This field is initialized to all 1s and can only be written once.

Register 12: User Register 1 (USER_REG1), offset 0x1E4

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

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

User Register 1 (USER_REG1)

Base 0x400F.E000

Offset 0x1E4

Type R/W, reset 0x8FFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NOTWRITTEN	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	0	0	0	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	NOTWRITTEN	R/W	1	Specifies that this 32-bit dword has not been written.
30:0	DATA	R/W	0xFFFFFFFF	Contains the user data value. This field is initialized to all 1s and can only be written once.

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

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

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

Flash Memory Protection Read Enable 1 (FMPRE1)

Base 0x400F.E000

Offset 0x204

Type R/W, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	READ_ENABLE															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	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
	READ_ENABLE															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

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

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

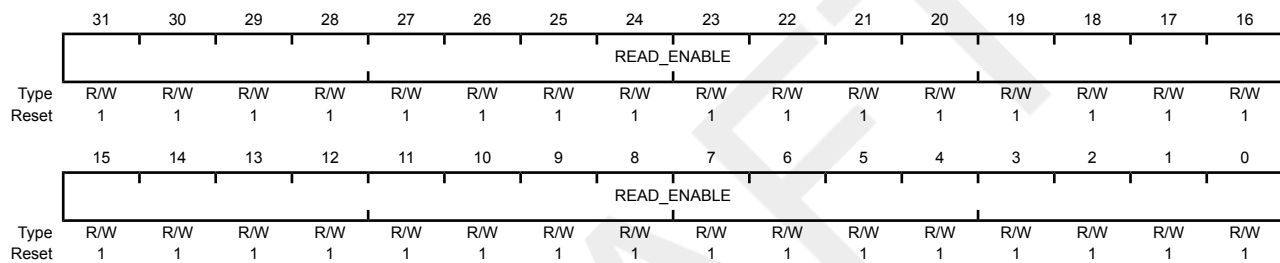
This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). This register is loaded during the power-on reset sequence. The factory settings for the **FMPREN** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves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

Flash Memory Protection Read Enable 2 (FMPRE2)

Base 0x400F.E000

Offset 0x208

Type R/W, reset 0xFFFF.FFFF



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

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

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

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

Flash Memory Protection Read Enable 3 (FMPRE3)

Base 0x400F.E000

Offset 0x20C

Type R/W, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	READ_ENABLE															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	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
	READ_ENABLE															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

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

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

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

Flash Memory Protection Program Enable 1 (FMPPE1)

Base 0x400F.E000

Offset 0x404

Type R/W, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PROG_ENABLE															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	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
	PROG_ENABLE															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

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

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

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

Flash Memory Protection Program Enable 2 (FMPPE2)

Base 0x400F.E000

Offset 0x408

Type R/W, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PROG_ENABLE															
Type	R/W	R/W	R/W	R/W	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
	PROG_ENABLE															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

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

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

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

Flash Memory Protection Program Enable 3 (FMPPE3)

Base 0x400F.E000

Offset 0x40C

Type R/W, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PROG_ENABLE															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	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
	PROG_ENABLE															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

9 General-Purpose Input/Outputs (GPIOs)

GPIO

The GPIO module is composed of seven physical GPIO blocks, each corresponding to an individual GPIO port (Port A, Port B, Port C, Port D, Port E, Port F, and Port G,). The GPIO module is FiRM-compliant and supports 3-38 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

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

9.1 Function Description

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

Each GPIO port is a separate hardware instantiation of the same physical block. The LM3S8938 microcontroller contains seven ports and thus seven of these physical GPIO blocks.

9.1.1 Data Control

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

9.1.1.1 Data Direction Operation

The **GPIO Direction (GPIODIR)** register (see page 163) is used to configure each individual pin as an input or output. When the data direction bit is set to 0, the GPIO is configured as an input and

the corresponding data register bit will capture and store the value on the GPIO port. When the data direction bit is set to 1, the GPIO is configured as an output and the corresponding data register bit will be driven out on the GPIO port.

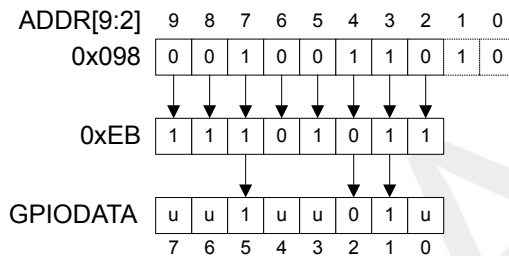
9.1.1.2 Data Register Operation

To aid in the efficiency of software, the GPIO ports allow for the modification of individual bits in the **GPIO Data (GPIODATA)** register (see page 162) by using bits [9:2] of the address bus as a mask. This allows software drivers to modify individual GPIO pins in a single instruction, without affecting the state of the other pins. This is in contrast to the "typical" method of doing a read-modify-write operation to set or clear an individual GPIO pin. To accommodate this feature, the **GPIODATA** register covers 256 locations in the memory map.

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

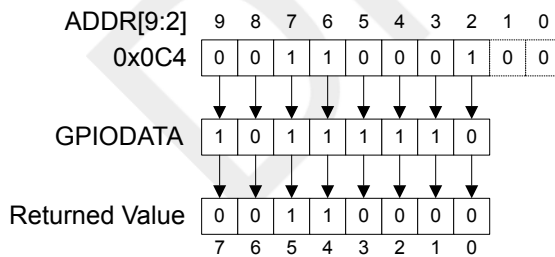
For example, writing a value of 0xEB to the address GPIODATA + 0x098 would yield as shown in Figure 9-1 on page 156, where u is data unchanged by the write.

Figure 9-1. GPIODATA Write Example



During a read, if the address bit associated with the data bit is set to 1, the value is read. If the address bit associated with the data bit is set to 0, it is read as a zero, regardless of its actual value. For example, reading address GPIODATA + 0x0C4 yields as shown in Figure 9-2 on page 156.

Figure 9-2. GPIODATA Read Example



9.1.2 Interrupt Control

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

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

- **GPIO Interrupt Sense (GPIOIS)** register (see page 164)
- **GPIO Interrupt Both Edges (GPIOIBE)** register (see page 165)
- **GPIO Interrupt Event (GPIOIEV)** register (see page 166)

Interrupts are enabled/disabled via the **GPIO Interrupt Mask (GPIOIM)** register (see page 167).

When an interrupt condition occurs, the state of the interrupt signal can be viewed in two locations: the **GPIO Raw Interrupt Status (GPIORIS)** and **GPIO Masked Interrupt Status (GPIOMIS)** registers (see page 168 and page 169). As the name implies, the **GPIOMIS** register only shows interrupt conditions that are allowed to be passed to the controller. The **GPIORIS** register indicates that a GPIO pin meets the conditions for an interrupt, but has not necessarily been sent to the controller.

In addition to providing GPIO functionality, PB4 can also be used as an external trigger for the ADC. If PB4 is configured as a non-masked interrupt pin (**GPIOIM** is set to 1), not only is an interrupt for PortB generated, but an external trigger signal is sent to the ADC. If the **ADC Event Multiplexer Select (ADCEMUX)** register is configured to use the external trigger, an ADC conversion is initiated.

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

Interrupts are cleared by writing a 1 to the **GPIO Interrupt Clear (GPIOICR)** register (see page 170).

When programming the following interrupt control registers, the interrupts should be masked (**GPIOIM** set to 0). Writing any value to an interrupt control register (**GPIOIS**, **GPIOIBE**, or **GPIOIEV**) can generate a spurious interrupt if the corresponding bits are enabled.

9.1.3 Mode Control

The GPIO pins can be controlled by either hardware or software. When hardware control is enabled via the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 171), the pin state is controlled by its alternate function (that is, the peripheral). Software control corresponds to GPIO mode, where the **GPIO DATA** register is used to read/write the corresponding pins.

9.1.4 Commit Control

The commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 171) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 181) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 182) have been set to 1.

9.1.5 Pad Control

The pad control registers allow for GPIO pad configuration by software based on the application requirements. The pad control registers include the **GPIO DR2R**, **GPIO DR4R**, **GPIO DR8R**, **GPIO DR**, **GPIO PUR**, **GPIO PDR**, **GPIO SLR**, and **GPIO DEN** registers.

9.1.6 Identification

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

9.2 Initialization and Configuration

To use the GPIO, the peripheral clock must be enabled by setting the appropriate GPIO Port bit field (**GPIO_n**) in the **RCGC2** register.

On reset, all GPIO pins (except for the five JTAG pins) are configured out of reset to be undriven (tristate): **GPIOAFSEL=0**, **GPIODEN=0**, **GPIOPDR=0**, and **GPIOPUR=0**. Table 9-1 on page 158 shows all possible configurations of the GPIO pads and the control register settings required to achieve them. Table 9-2 on page 158 shows how a rising edge interrupt would be configured for pin 2 of a GPIO port.

Table 9-1. GPIO Pad Configuration Examples

Configuration	GPIO Register Bit Value ^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 Input (GPIO)	0	0	1	1	X	X	X	X	X	X
Open Drain Output (GPIO)	0	1	1	1	X	X	?	?	?	?
Open Drain Input/Output (I ² C)	1	X	1	1	X	X	?	?	?	?
Digital Input (Timer CCP)	1	X	0	1	?	?	X	X	X	X
Digital Output (Timer PWM)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (SSI)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (UART)	1	X	0	1	?	?	?	?	?	?
Analog Input (Comparator)	0	0	0	0	0	0	X	X	X	X
Digital Output (Comparator)	1	X	0	1	?	?	?	?	?	?

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

?=Can be either 0 or 1, depending on the configuration

Table 9-2. GPIO Interrupt Configuration Example

Register	Desired Interrupt Event Trigger	Pin 2 Bit Value ^a							
		7	6	5	4	3	2	1	0
GPIOIS	0=edge	X	X	X	X	X	0	X	X
	1=level								

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

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

9.3 Register Map

Table 9-3 on page 160 lists the GPIO registers. The offset listed is a hexadecimal increment to the register's address, relative to that GPIO port's base address:

- GPIO Port A: 0x4000.4000
- GPIO Port B: 0x4000.5000
- GPIO Port C: 0x4000.6000
- GPIO Port D: 0x4000.7000
- GPIO Port E: 0x4002.4000
- GPIO Port F: 0x4002.5000
- GPIO Port G: 0x4002.6000

Important: The GPIO registers in this chapter are duplicated in each GPIO block, however, depending on the block, all eight bits may not be connected to a GPIO pad. In those cases, writing to those unconnected bits has no effect and reading those unconnected bits returns no meaningful data.

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

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

The default reset value for the **GPIOCR** register is 0x0000.00FF for all GPIO pins, with the exception of the five JTAG/SWD pins (PB7 and PC[3:0]). To ensure that the JTAG port is not accidentally programmed as a GPIO, these five pins default to non-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 9-3. GPIO Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPIODATA	R/W	0x0000.0000	GPIO Data	162
0x400	GPIODIR	R/W	0x0000.0000	GPIO Direction	163
0x404	GPIOIS	R/W	0x0000.0000	GPIO Interrupt Sense	164
0x408	GPIOIBE	R/W	0x0000.0000	GPIO Interrupt Both Edges	165
0x40C	GPIOIEV	R/W	0x0000.0000	GPIO Interrupt Event	166
0x410	GPIOIM	R/W	0x0000.0000	GPIO Interrupt Mask	167
0x414	GPIORIS	RO	0x0000.0000	GPIO Raw Interrupt Status	168
0x418	GIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status	169
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear	170
0x420	GPIOAFSEL	R/W	-	GPIO Alternate Function Select	171
0x500	GPIODR2R	R/W	0x0000.00FF	GPIO 2-mA Drive Select	173
0x504	GPIODR4R	R/W	0x0000.0000	GPIO 4-mA Drive Select	174
0x508	GPIODR8R	R/W	0x0000.0000	GPIO 8-mA Drive Select	175
0x50C	GPIOODR	R/W	0x0000.0000	GPIO Open Drain Select	176
0x510	GIOPUR	R/W	-	GPIO Pull-Up Select	177
0x514	GIOPDR	R/W	0x0000.0000	GPIO Pull-Down Select	178
0x518	GPIOSLR	R/W	0x0000.0000	GPIO Slew Rate Control Select	179
0x51C	GPIODEN	R/W	-	GPIO Digital Enable	180
0x520	GPIOLOCK	R/W	0x0000.0001	GPIO Lock	181
0x524	GPIOCR	-	-	GPIO Commit	182
0xFD0	GPIOPeriphID4	RO	0x0x0000.0000	GPIO Peripheral Identification 4	184
0xFD4	GPIOPeriphID5	RO	0x0x0000.0000	GPIO Peripheral Identification 5	185
0xFD8	GPIOPeriphID6	RO	0x0x0000.0000	GPIO Peripheral Identification 6	186
0xFDC	GPIOPeriphID7	RO	0x0x0000.0000	GPIO Peripheral Identification 7	187
0xFE0	GPIOPeriphID0	RO	0x0x0000.0061	GPIO Peripheral Identification 0	188
0xFE4	GPIOPeriphID1	RO	0x0x0000.0000	GPIO Peripheral Identification 1	189
0xFE8	GPIOPeriphID2	RO	0x0x0000.0018	GPIO Peripheral Identification 2	190
0xFEC	GPIOPeriphID3	RO	0x0x0000.0001	GPIO Peripheral Identification 3	191

Offset	Name	Type	Reset	Description	See page
0xFF0	GPIOCellID0	RO	0x0x0000.000D	GPIO PrimeCell Identification 0	192
0xFF4	GPIOCellID1	RO	0x0x0000.00F0	GPIO PrimeCell Identification 1	193
0xFF8	GPIOCellID2	RO	0x0x0000.0005	GPIO PrimeCell Identification 2	194
0xFFC	GPIOCellID3	RO	0x0x0000.00B1	GPIO PrimeCell Identification 3	195

9.4 Register Descriptions

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

DRAFT

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

The **GPIODATA** register is the data register. In software control mode, values written in the **GPIODATA** register are transferred onto the GPIO port pins if the respective pins have been configured as outputs through the **GPIO Direction (GPIODIR)** register (see page 163).

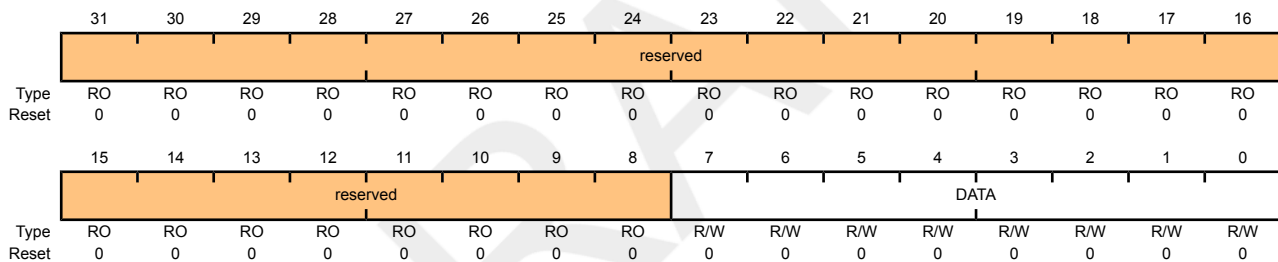
In order to write to **GPIODATA**, the corresponding bits in the mask, resulting from the address bus bits [9:2], must be High. Otherwise, the bit values remain unchanged by the write.

Similarly, the values read from this register are determined for each bit by the mask bit derived from the address used to access the data register, bits [9:2]. Bits that are 1 in the address mask cause the corresponding bits in **GPIODATA** to be read, and bits that are 0 in the address mask cause the corresponding bits in **GPIODATA** to be read as 0, regardless of their value.

A read from **GPIODATA** returns the last bit value written if the respective pins are configured as outputs, or it returns the value on the corresponding input pin when these are configured as inputs. All bits are cleared by a reset.

GPIO Data (GPIODATA)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0x000
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved 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	0	GPIO Data

This register is virtually mapped to 256 locations in the address space. To facilitate the reading and writing of data to these registers by independent drivers, the data read from and the data written to the registers are masked by the eight address lines `ipaddr[9:2]`. Reads from this register return its current state. Writes to this register only affect bits that are not masked by `ipaddr[9:2]` and are configured as outputs. See "Data Register Operation" on page 156 for examples of reads and writes.

Register 2: GPIO Direction (GPIODIR), offset 0x400

The **GPIODIR** register is the data direction register. Bits set to 1 in the **GPIODIR** register configure the corresponding pin to be an output, while bits set to 0 configure the pins to be inputs. All bits are cleared by a reset, meaning all GPIO pins are inputs by default.

GPIO Direction (GPIODIR)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0x400
 Type R/W, reset 0x0000.0000

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

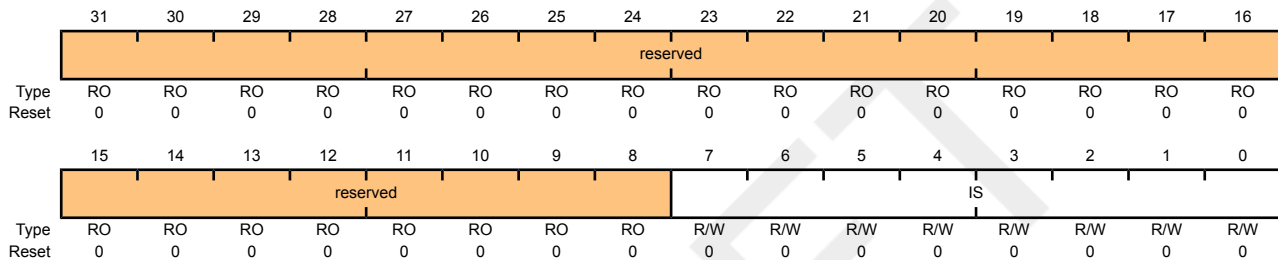
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIR	R/W	0x00	GPIO Data Direction 0: Pins are inputs. 1: Pins are outputs.

Register 3: GPIO Interrupt Sense (GPIOIS), offset 0x404

The **GPIOIS** register is the interrupt sense register. Bits set to 1 in **GPIOIS** configure the corresponding pins to detect levels, while bits set to 0 configure the pins to detect edges. All bits are cleared by a reset.

GPIO Interrupt Sense (GPIOIS)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0x404
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved 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 0: Edge on corresponding pin is detected (edge-sensitive). 1: Level on corresponding pin is detected (level-sensitive).

Register 4: GPIO Interrupt Both Edges (GPIOIBE), offset 0x408

The **GPIOIBE** register is the interrupt both-edges register. When the corresponding bit in the **GPIO Interrupt Sense (GPIOIS)** register (see page 164) is set to detect edges, bits set to High in **GPIOIBE** configure the corresponding pin to detect both rising and falling edges, regardless of the corresponding bit in the **GPIO Interrupt Event (GPIOIEV)** register (see page 166). Clearing a bit configures the pin to be controlled by **GPIOIEV**. All bits are cleared by a reset.

GPIO Interrupt Both Edges (GPIOIBE)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0x408

Type R/W, reset 0x0000.0000

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

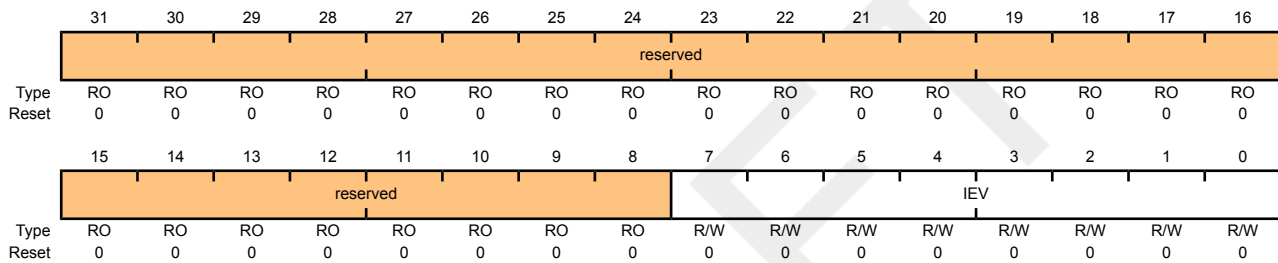
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IBE	R/W	0x00	GPIO Interrupt Both Edges 0: Interrupt generation is controlled by the GPIO Interrupt Event (GPIOIEV) register (see page 142). 1: Both edges on the corresponding pin trigger an interrupt. Note: Single edge is determined by the corresponding bit in GPIOIEV .

Register 5: GPIO Interrupt Event (GPIOIEV), offset 0x40C

The **GPIOIEV** register is the interrupt event register. Bits set to High in **GPIOIEV** configure the corresponding pin to detect rising edges or high levels, depending on the corresponding bit value in the **GPIO Interrupt Sense (GPIOIS)** register (see page 164). Clearing a bit configures the pin to detect falling edges or low levels, depending on the corresponding bit value in **GPIOIS**. All bits are cleared by a reset.

GPIO Interrupt Event (GPIOIEV)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0x40C
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved 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 0: Falling edge or Low levels on corresponding pins trigger interrupts. 1: Rising edge or High levels on corresponding pins trigger interrupts.

Register 6: GPIO Interrupt Mask (GPIOIM), offset 0x410

The **GPIOIM** register is the interrupt mask register. Bits set to High in **GPIOIM** allow the corresponding pins to trigger their individual interrupts and the combined GPIOINTR line. Clearing a bit disables interrupt triggering on that pin. All bits are cleared by a reset.

GPIO Interrupt Mask (GPIOIM)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0x410
 Type R/W, reset 0x0000.0000

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

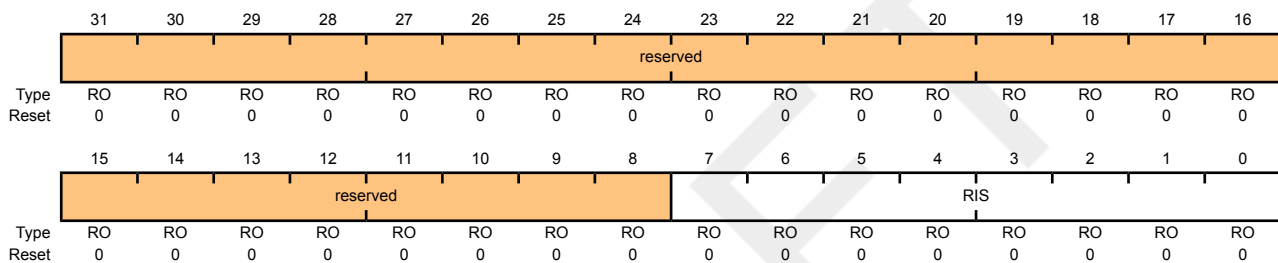
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IME	R/W	0x00	GPIO Interrupt Mask Enable 0: Corresponding pin interrupt is masked. 1: Corresponding pin interrupt is not masked.

Register 7: GPIO Raw Interrupt Status (GPIORIS), offset 0x414

The **GPIORIS** register is the raw interrupt status register. Bits read High in **GPIORIS** reflect the status of interrupt trigger conditions detected (raw, prior to masking), indicating that all the requirements have been met, before they are finally allowed to trigger by the **GPIO Interrupt Mask (GPIOIM)** register (see page 167). Bits read as zero indicate that corresponding input pins have not initiated an interrupt. All bits are cleared by a reset.

GPIO Raw Interrupt Status (GPIORIS)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0x414
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved 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 Reflect the status of interrupt trigger condition detection on pins (raw, prior to masking). 0: Corresponding pin interrupt requirements not met. 1: Corresponding pin interrupt has met requirements.

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

The **GPIOMIS** register is the masked interrupt status register. Bits read High in **GPIOMIS** reflect the status of input lines triggering an interrupt. Bits read as Low indicate that either no interrupt has been generated, or the interrupt is masked.

In addition to providing GPIO functionality, PB4 can also be used as an external trigger for the ADC. If PB4 is configured as a non-masked interrupt pin (`GPIOIM` is set to 1), not only is an interrupt for PortB generated, but an external trigger signal is sent to the ADC. If the **ADC Event Multiplexer Select (ADCEMUX)** register is configured to use the external trigger, an ADC conversion is initiated.

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

GPIOMIS is the state of the interrupt after masking.

GPIO Masked Interrupt Status (GPIOMIS)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0x418
 Type RO, reset 0x0000.0000

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

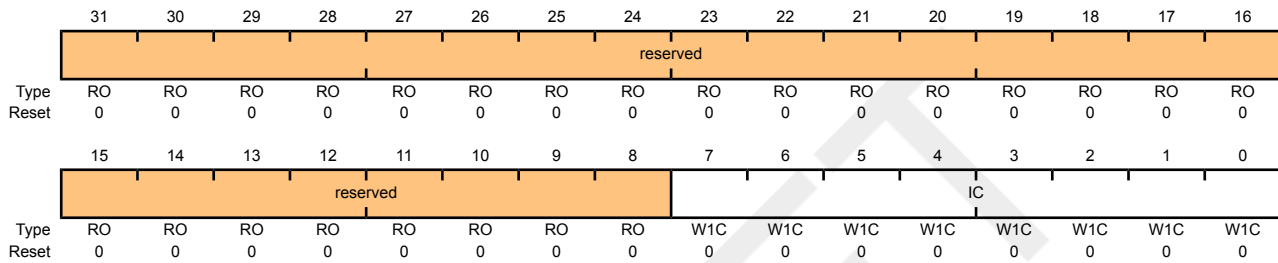
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	MIS	RO	0x00	GPIO Masked Interrupt Status Masked value of interrupt due to corresponding pin. 0: Corresponding GPIO line interrupt not active. 1: Corresponding GPIO line asserting interrupt.

Register 9: GPIO Interrupt Clear (GPIOICR), offset 0x41C

The **GPIOICR** register is the interrupt clear register. Writing a 1 to a bit in this register clears the corresponding interrupt edge detection logic register. Writing a 0 has no effect.

GPIO Interrupt Clear (GPIOICR)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0x41C
 Type W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved 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 0: Corresponding interrupt is unaffected. 1: Corresponding interrupt is cleared.

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

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

The commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 171) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 181) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 182) have been set to 1.

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

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

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

GPIO Alternate Function Select (GPIOAFSEL)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0x420

Type R/W, reset -

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

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

Bit/Field	Name	Type	Reset	Description
7:0	AFSEL	R/W	-	<p>GPIO Alternate Function Select</p> <p>0: Software control of corresponding GPIO line (GPIO mode).</p> <p>1: Hardware control of corresponding GPIO line (alternate hardware function).</p> <p>Note: The default reset value for the GPIOAFSEL, GPIOPUR, and GIODEN registers are 0x0000.0000 for all GPIO pins, with the exception of the five JTAG/SWD pins (PB7 and PC[3:0]). These five pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers for GPIO Port B is 0x0000.0080 while the default reset value for Port C is 0x0000.000F.</p>

DRAFT

Register 11: GPIO 2-mA Drive Select (GPIODR2R), offset 0x500

The **GPIODR2R** register is the 2-mA drive control register. It allows for each GPIO signal in the port to be individually configured without affecting the other pads. When writing a **DRV2** bit for a GPIO signal, the corresponding **DRV4** bit in the **GPIODR4R** register and the **DRV8** bit in the **GPIODR8R** register are automatically cleared by hardware.

GPIO 2-mA Drive Select (GPIODR2R)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0x500

Type R/W, reset 0x0000.00FF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DRV2							
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	1	1	1	1	1	1	1	1

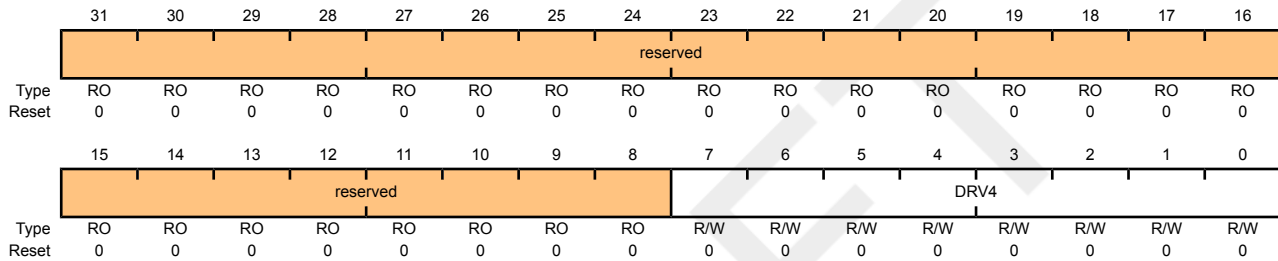
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV2	R/W	0xFF	Output Pad 2-mA Drive Enable A write of 1 to either GPIODR4[n] or GPIODR8[n] clears the corresponding 2-mA enable bit. The change is effective on the second clock cycle after the write.

Register 12: GPIO 4-mA Drive Select (GPIODR4R), offset 0x504

The **GPIODR4R** register is the 4-mA drive control register. It allows for each GPIO signal in the port to be individually configured without affecting the other pads. When writing the **DRV4** bit for a GPIO signal, the corresponding **DRV2** bit in the **GPIODR2R** register and the **DRV8** bit in the **GPIODR8R** register are automatically cleared by hardware.

GPIO 4-mA Drive Select (GPIODR4R)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0x504
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV4	R/W	0x00	Output Pad 4-mA Drive Enable A write of 1 to either GPIODR2[n] or GPIODR8[n] clears the corresponding 4-mA enable bit. The change is effective on the second clock cycle after the write.

Register 13: GPIO 8-mA Drive Select (GPIODR8R), offset 0x508

The **GPIODR8R** register is the 8-mA drive control register. It allows for each GPIO signal in the port to be individually configured without affecting the other pads. When writing the **DRV8** bit for a GPIO signal, the corresponding **DRV2** bit in the **GPIODR2R** register and the **DRV4** bit in the **GPIODR4R** register are automatically cleared by hardware.

GPIO 8-mA Drive Select (GPIODR8R)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0x508

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DRV8							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV8	R/W	0x00	Output Pad 8-mA Drive Enable A write of 1 to either GPIODR2[n] or GPIODR4[n] clears the corresponding 8-mA enable bit. The change is effective on the second clock cycle after the write.

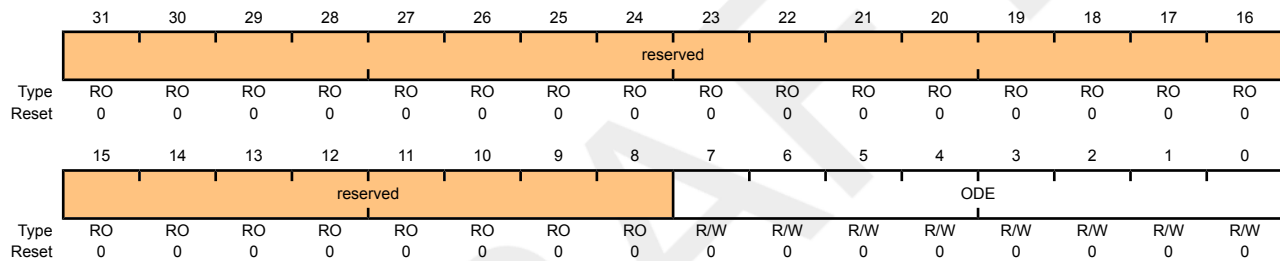
Register 14: GPIO Open Drain Select (GPIOODR), offset 0x50C

The **GPIOODR** register is the open drain control register. Setting a bit in this register enables the open drain configuration of the corresponding GPIO pad. When open drain mode is enabled, the corresponding bit should also be set in the **GPIO Digital Input Enable (GPIODEN)** register (see page 180). Corresponding bits in the drive strength registers (**GPIODR2R**, **GPIODR4R**, **GPIODR8R**, and **GPIOSLR**) can be set to achieve the desired rise and fall times. The GPIO acts as an open drain input if the corresponding bit in the **GPIODIR** register is set to 0; and as an open drain output when set to 1.

When using the I²C module, the **GPIO Alternate Function Select (GPIOAFSEL)** register bit for PB2 and PB3 should be set to 1 (see examples in “Initialization and Configuration” on page 158).

GPIO Open Drain Select (GPIOODR)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0x50C
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved 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 0: Open drain configuration is disabled. 1: Open drain configuration is enabled.

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

The **GPIOPUR** register is the pull-up control register. When a bit is set to 1, it enables a weak pull-up resistor on the corresponding GPIO signal. Setting a bit in **GPIOPUR** automatically clears the corresponding bit in the **GPIO Pull-Down Select (GPIOPDR)** register (see page 178).

GPIO Pull-Up Select (GPIOPUR)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0x510
 Type R/W, reset -

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

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

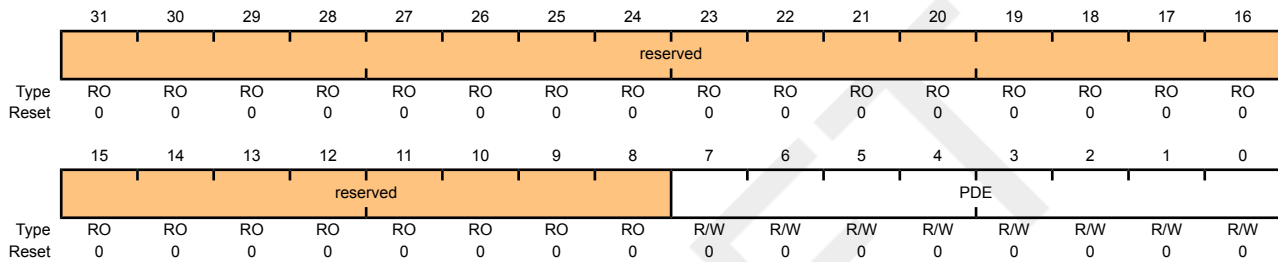
Note: The default reset value for the **GPIOAFSEL**, **GPIOPUR**, and **GPIODEN** registers are 0x0000.0000 for all GPIO pins, with the exception of the five JTAG/SWD pins (**PB7** and **PC[3:0]**). These five pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers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

Register 16: GPIO Pull-Down Select (GPIOPDR), offset 0x514

The **GPIOPDR** register is the pull-down control register. When a bit is set to 1, it enables a weak pull-down resistor on the corresponding GPIO signal. Setting a bit in **GPIOPDR** automatically clears the corresponding bit in the **GPIO Pull-Up Select (GPIOPUR)** register (see page 177).

GPIO Pull-Down Select (GPIOPDR)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0x514
 Type R/W, reset 0x0000.0000



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

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

The **GPIOSLR** register is the slew rate control register. Slew rate control is only available when using the 8-mA drive strength option via the **GPIO 8-mA Drive Select (GPIODR8R)** register (see page 175).

GPIO Slew Rate Control Select (GPIOSLR)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0x518
 Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved 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	0	Slew Rate Limit Enable (8-mA drive only) 0: Slew rate control disabled. 1: Slew rate control enabled.

Register 18: GPIO Digital Enable (GPIODEN), offset 0x51C

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

GPIO Digital Enable (GPIODEN)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0x51C
 Type R/W, reset -

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved 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 0: Digital functions disabled. 1: Digital functions enabled.

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

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

The **GPIOLOCK** register enables write access to the **GPIOCR** register (see page 182). Writing 0x1ACCE551 to the **GPIOLOCK** register will unlock the **GPIOCR** register. Writing any other value to the **GPIOLOCK** register re-enables the locked state. Reading the **GPIOLOCK** register returns the lock status rather than the 32-bit value that was previously written. Therefore, when write accesses are disabled, or locked, reading the **GPIOLOCK** register returns 0x00000001. When write accesses are enabled, or unlocked, reading the **GPIOLOCK** register returns 0x00000000.

GPIO Lock (GPIOLOCK)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0x520

Type R/W, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	LOCK															
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
	LOCK															
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	1

Bit/Field	Name	Type	Reset	Description
31:0	LOCK	R/W	0x00000001	GPIO Lock

A write of the value 0x1ACCE551 unlocks the GPIO Commit register for write access. A write of any other value reapplies the lock, preventing any register updates. A read of this register returns the following values:

locked: 0x00000001

unlocked: 0x00000000

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

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

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

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

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

GPIO Commit (GPIOCR)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0x524
 Type -, reset -

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

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

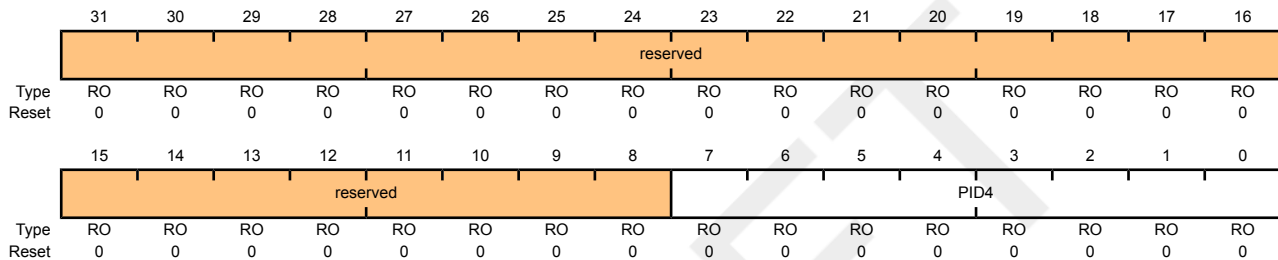
Bit/Field	Name	Type	Reset	Description
7:0	CR	-	-	<p>GPIO Commit</p> <p>On a bit-wise basis, any bit set allows the corresponding <code>GPIOAFSEL</code> bit to be set to its alternate function.</p> <p>Note: The default register type for the GPIOCR register is RO for all GPIO pins, with the exception of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

Register 21: GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0

The **GPIOPeriphID4**, **GPIOPeriphID5**, **GPIOPeriphID6**, and **GPIOPeriphID7** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 4 (GPIOPeriphID4)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0xFD0
 Type RO, reset 0x0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	GPIO Peripheral ID Register[7:0]

Register 22: GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4

The **GPIOPeriphID4**, **GPIOPeriphID5**, **GPIOPeriphID6**, and **GPIOPeriphID7** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 5 (GPIOPeriphID5)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

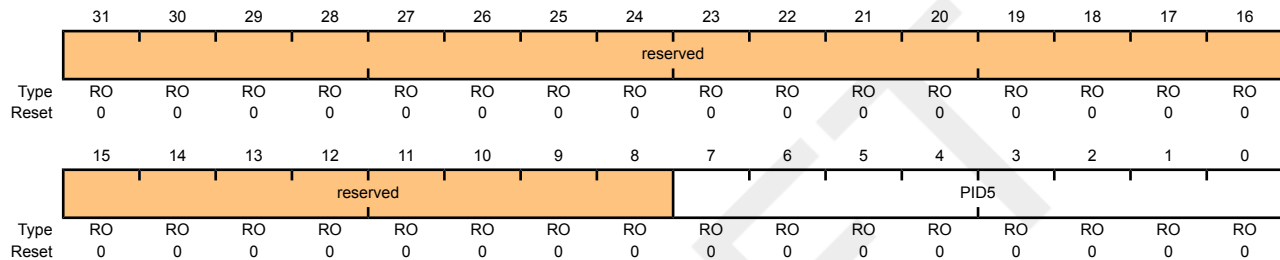
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0xFD4

Type RO, reset 0x0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	GPIO Peripheral ID Register[15:8]

Register 23: GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8

The **GPIOPeriphID4**, **GPIOPeriphID5**, **GPIOPeriphID6**, and **GPIOPeriphID7** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 6 (GPIOPeriphID6)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0xFD8
 Type RO, reset 0x0x0000.0000



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

Register 24: GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC

The **GPIOPeriphID4**, **GPIOPeriphID5**, **GPIOPeriphID6**, and **GPIOPeriphID7** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 7 (GPIOPeriphID7)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

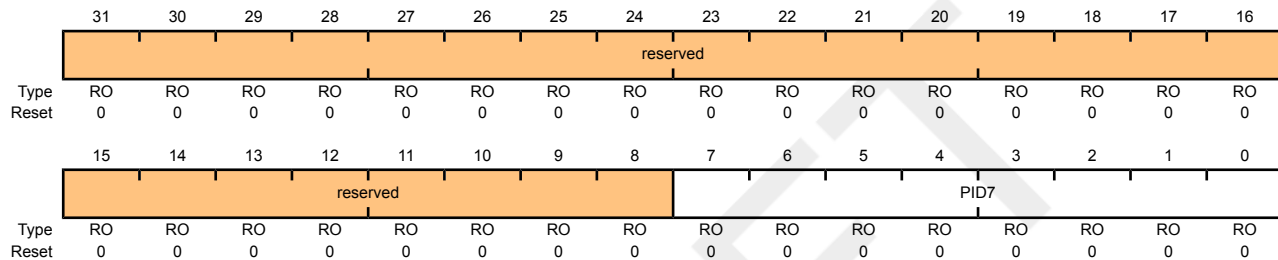
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0xFDC

Type RO, reset 0x0x0000.0000



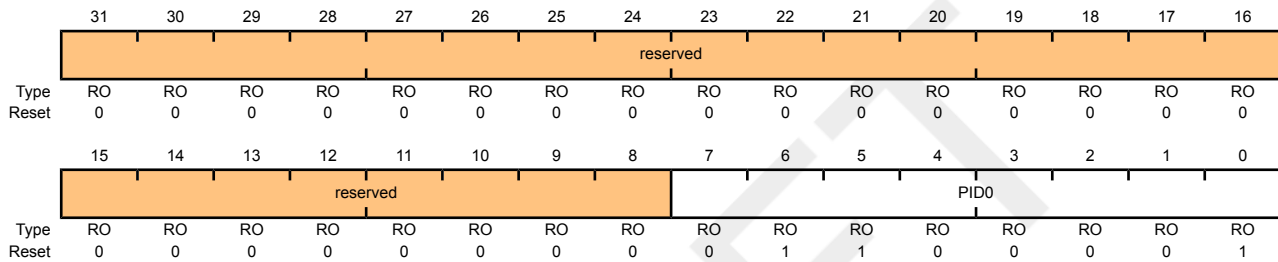
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	GPIO Peripheral ID Register[31:24]

Register 25: GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0

The **GPIOPeriphID0**, **GPIOPeriphID1**, **GPIOPeriphID2**, and **GPIOPeriphID3** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 0 (GPIOPeriphID0)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0xFE0
 Type RO, reset 0x0x0000.0061



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x61	GPIO Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 26: GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4

The **GPIOPeriphID0**, **GPIOPeriphID1**, **GPIOPeriphID2**, and **GPIOPeriphID3** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 1 (GPIOPeriphID1)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

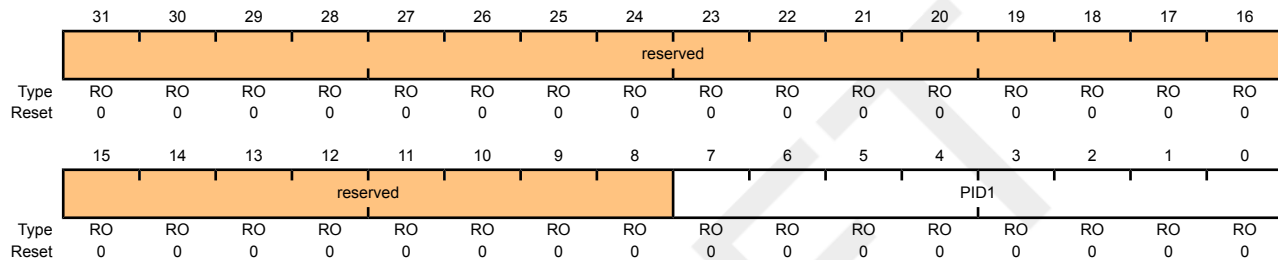
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0xFE4

Type RO, reset 0x0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	GPIO Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

Register 27: GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8

The **GPIOPeriphID0**, **GPIOPeriphID1**, **GPIOPeriphID2**, and **GPIOPeriphID3** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 2 (GPIOPeriphID2)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0xFE8
 Type RO, reset 0x0x0000.0018



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	GPIO Peripheral ID Register[23:16] Can be used by software to identify the presence of this peripheral.

Register 28: GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC

The **GPIOPeriphID0**, **GPIOPeriphID1**, **GPIOPeriphID2**, and **GPIOPeriphID3** registers can conceptually be treated as one 32-bit register; each register contains eight bits of the 32-bit register, used by software to identify the peripheral.

GPIO Peripheral Identification 3 (GPIOPeriphID3)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

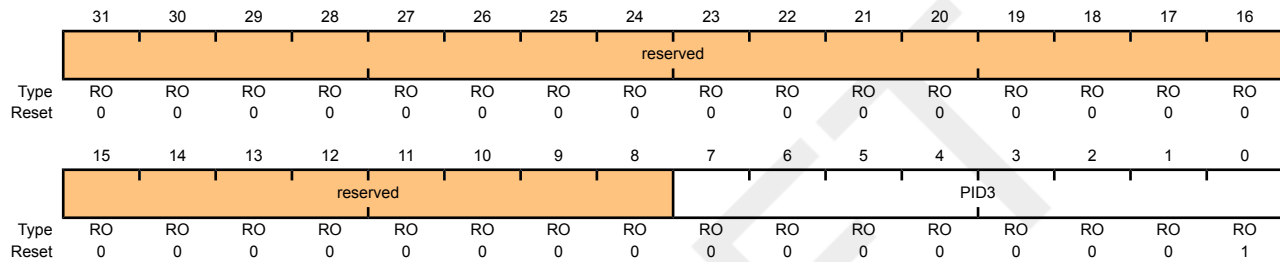
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0xFEC

Type RO, reset 0x0x0000.0001



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

Register 29: GPIO PrimeCell Identification 0 (GPIOCellID0), offset 0xFF0

The **GPIOCellID0**, **GPIOCellID1**, **GPIOCellID2**, and **GPIOCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIO PrimeCell Identification 0 (GPIOCellID0)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0xFF0
 Type RO, reset 0x0x0000.000D



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

Register 30: GPIO PrimeCell Identification 1 (GPIOCellID1), offset 0xFF4

The **GPIOCellID0**, **GPIOCellID1**, **GPIOCellID2**, and **GPIOCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIO PrimeCell Identification 1 (GPIOCellID1)

GPIO Port A base: 0x4000.4000

GPIO Port B base: 0x4000.5000

GPIO Port C base: 0x4000.6000

GPIO Port D base: 0x4000.7000

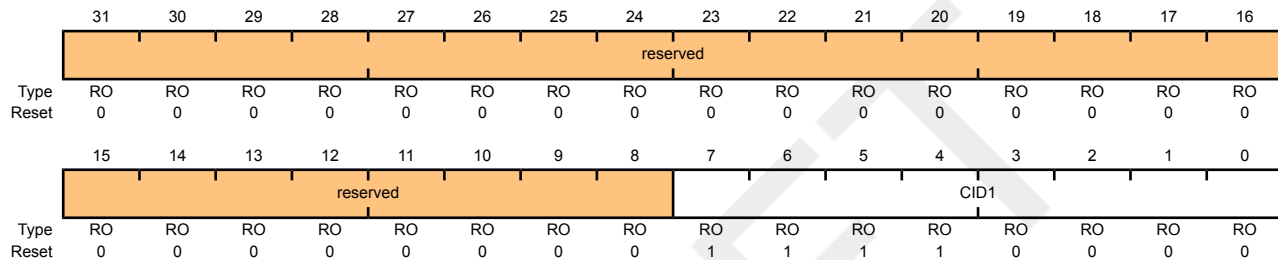
GPIO Port E base: 0x4002.4000

GPIO Port F base: 0x4002.5000

GPIO Port G base: 0x4002.6000

Offset 0xFF4

Type RO, reset 0x0x0000.00F0



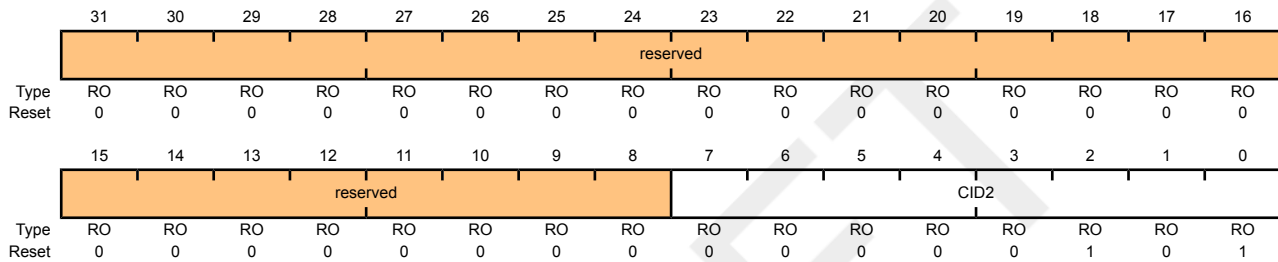
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	GPIO PrimeCell ID Register[15:8] Provides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

Register 31: GPIO PrimeCell Identification 2 (GPIOCellID2), offset 0xFF8

The **GPIOCellID0**, **GPIOCellID1**, **GPIOCellID2**, and **GPIOCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIO PrimeCell Identification 2 (GPIOCellID2)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0xFF8
 Type RO, reset 0x0x0000.0005



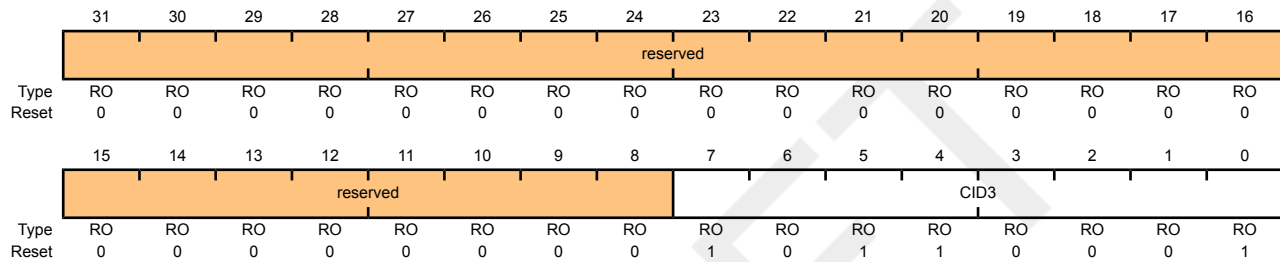
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	GPIO PrimeCell ID Register[23:16] Provides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

Register 32: GPIO PrimeCell Identification 3 (GPIOCellID3), offset 0xFFC

The **GPIOCellID0**, **GPIOCellID1**, **GPIOCellID2**, and **GPIOCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIO PrimeCell Identification 3 (GPIOCellID3)

GPIO Port A base: 0x4000.4000
 GPIO Port B base: 0x4000.5000
 GPIO Port C base: 0x4000.6000
 GPIO Port D base: 0x4000.7000
 GPIO Port E base: 0x4002.4000
 GPIO Port F base: 0x4002.5000
 GPIO Port G base: 0x4002.6000
 Offset 0xFFC
 Type RO, reset 0x0x0000.00B1



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	GPIO PrimeCell ID Register[31:24] Provides software a standard cross-peripheral identification system.

10 General-Purpose Timers

GPTM

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

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

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

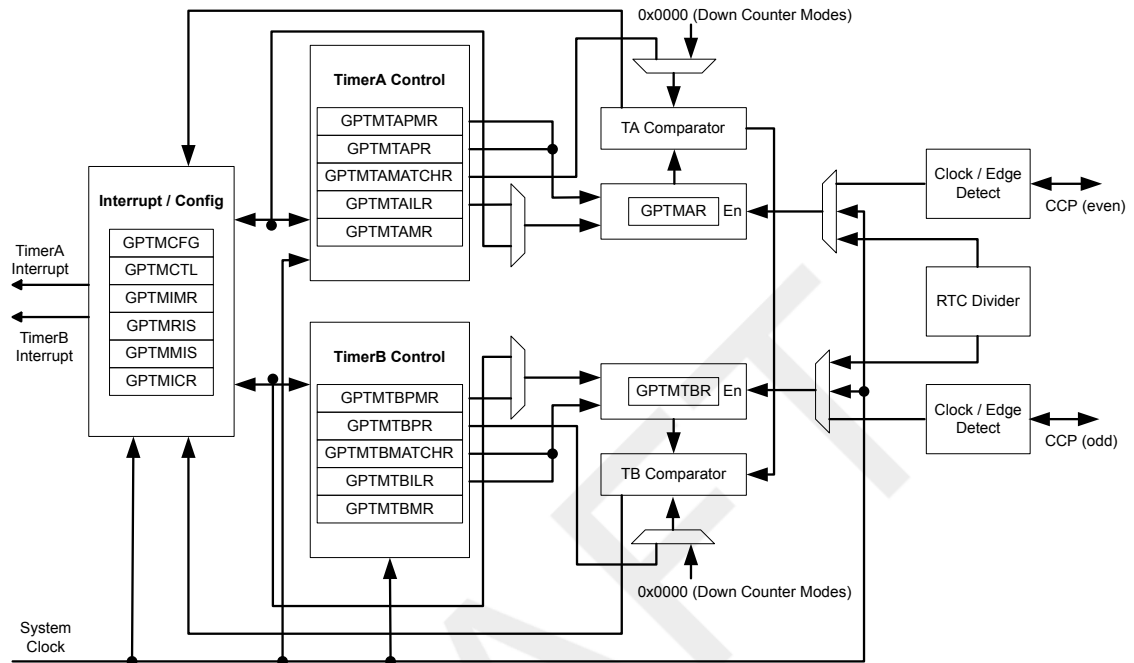
The General-Purpose Timer Module is one timing resource available on the Stellaris[®] microcontrollers. Other timer resources include the System Timer (SysTick) (see “System Timer (SysTick)” on page 38).

The following modes are supported:

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

10.1 Block Diagram

Figure 10-1. GPTM Module Block Diagram



10.2 Functional Description

The main components of each GPTM block are two free-running 16-bit up/down counters (referred to as TimerA and TimerB), two 16-bit match registers, two prescaler match registers, and two 16-bit load/initialization registers and their associated control functions. The exact functionality of each GPTM is controlled by software and configured through the register interface.

Software configures the GPTM using the **GPTM Configuration (GPTMCFG)** register (see page 208), the **GPTM TimerA Mode (GPTMTAMR)** register (see page 209), and the **GPTM TimerB Mode (GPTMTBMR)** register (see page 210). When in one of the 32-bit modes, the timer can only act as a 32-bit timer. However, when configured in 16-bit mode, the GPTM can have its two 16-bit timers configured in any combination of the 16-bit modes.

10.2.1 GPTM Reset Conditions

After reset has been applied to the GPTM module, the module is in an inactive state, and all control registers are cleared and in their default states. Counters TimerA and TimerB are initialized to 0xFFFF, along with their corresponding load registers: the **GPTM TimerA Interval Load (GPTMTAILR)** register (see page 219) and the **GPTM TimerB Interval Load (GPTMTBILR)** register (see page 220). The prescale counters are initialized to 0x00: the **GPTM TimerA Prescale (GPTMTAPR)** register (see page 223) and the **GPTM TimerB Prescale (GPTMTBPR)** register (see page 224).

10.2.2 32-Bit Timer Operating Modes

Note: Both the odd- and even-numbered CCP pins are used for 16-bit mode. Only the even-numbered CCP pins are used for 32-bit mode.

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

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

- **GPTM TimerA Interval Load (GPTMTAILR)** register [15:0], see page 219
- **GPTM TimerB Interval Load (GPTMTBILR)** register [15:0], see page 220
- **GPTM TimerA (GPTMTAR)** register [15:0], see page 227
- **GPTM TimerB (GPTMTBR)** register [15:0], see page 228

In the 32-bit modes, the GPTM translates a 32-bit write access to **GPTMTAILR** into a write access to both **GPTMTAILR** and **GPTMTBILR**. The resulting word ordering for such a write operation is:

```
GPTMTBILR[15:0]:GPTMTAILR[15:0]
```

Likewise, a read access to **GPTMTAR** returns the value:

```
GPTMTBR[15:0]:GPTMTAR[15:0]
```

10.2.2.1 32-Bit One-Shot/Periodic Timer Mode

In 32-bit one-shot and periodic timer modes, the concatenated versions of the TimerA and TimerB registers are configured as a 32-bit down-counter. The selection of one-shot or periodic mode is determined by the value written to the **TAMR** field of the **GPTM TimerA Mode (GPTMTAMR)** register (see page 209), and there is no need to write to the GPTM TimerB Mode (GPTMTBMR) register.

When software writes the **TAEN** bit in the **GPTM Control (GPTMCTL)** register (see page 211), the timer begins counting down from its preloaded value. Once the 0x0000.0000 state is reached, the timer reloads its start value from the concatenated **GPTMTAILR** on the next cycle. If configured to be a one-shot timer, the timer stops counting and clears the **TAEN** bit in the **GPTMCTL** register. If configured as a periodic timer, it continues counting.

In addition to reloading the count value, the GPTM generates interrupts and output triggers when it reaches the 0x00000000 state. The GPTM sets the **TATORIS** bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register (see page 215), and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register (see page 217). If the time-out interrupt is enabled in the **GPTM Interrupt Mask (GPTIMR)** register (see page 213), the GPTM also sets the **TATOMIS** bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register (see page 216).

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

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

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

10.2.2.2 32-Bit Real-Time Clock Timer Mode

In Real-Time Clock (RTC) mode, the concatenated versions of the TimerA and TimerB registers are configured as a 32-bit up-counter. When RTC mode is selected for the first time, the counter is

loaded with a value of 0x0000.0001. All subsequent load values must be written to the **GPTM TimerA Match (GPTMTAMATCHR)** register (see page 221) by the controller.

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

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

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

10.2.3 16-Bit Timer Operating Modes

The GPTM is placed into global 16-bit mode by writing a value of 0x4 to the **GPTM Configuration (GPTMCFG)** register (see page 208). This section describes each of the GPTM 16-bit modes of operation. TimerA and TimerB have identical modes, so a single description is given using an *n* to reference both.

10.2.3.1 16-Bit One-Shot/Periodic Timer Mode

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

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

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

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

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

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

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

Table 10-1. 16-Bit Timer With Prescaler Configurations

Prescale	#Clock (T c) ^a	Max Time	Units
00000000	1	1.3107	mS
00000001	2	2.6214	mS
00000010	3	23.9321	mS
-----	--	--	--
11111100	254	332.9229	mS
11111110	255	334.2336	mS
11111111	256	335.5443	mS

a. T_c is the clock period.

10.2.3.2 16-Bit Input Edge Count Mode

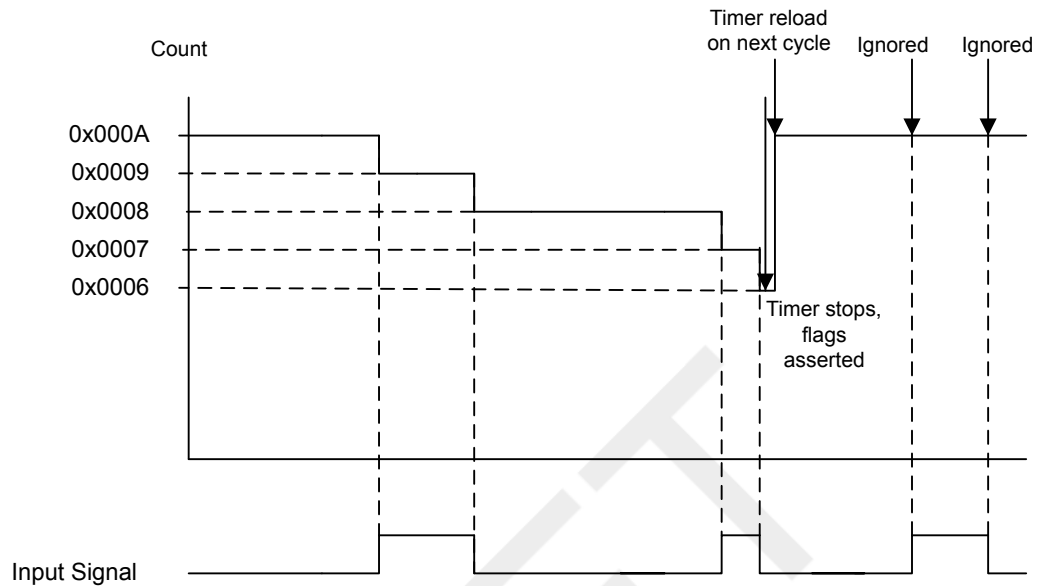
In Edge Count mode, the timer is configured as a down-counter capable of capturing three types of events: rising edge, falling edge, or both. To place the timer in Edge Count mode, the T_nCMR bit of the **GPTMTnMR** register must be set to 0. The type of edge that the timer counts is determined by the T_nEVENT fields of the **GPTMCTL** register. During initialization, the **GPTM Timern Match (GPTMTnMATCHR)** register is configured so that the difference between the value in the **GPTMTnILR** register and the **GPTMTnMATCHR** register equals the number of edge events that must be counted.

When software writes the T_nEN bit in the **GPTM Control (GPTMCTL)** register, the timer is enabled for event capture. Each input event on the CCP pin decrements the counter by 1 until the event count matches **GPTMTnMATCHR**. When the counts match, the GPTM asserts the C_nMRIS bit in the **GPTMRIS** register (and the C_nMMIS bit, if the interrupt is not masked). The counter is then reloaded using the value in **GPTMTnILR**, and stopped since the GPTM automatically clears the T_nEN bit in the **GPTMCTL** register. Once the event count has been reached, all further events are ignored until T_nEN is re-enabled by software.

Figure 10-2 on page 201 shows how input edge count mode works. In this case, the timer start value is set to **GPTMTnILR** = 0x000A and the match value is set to **GPTMTnMATCHR** = 0x0006 so that four edge events are counted. The counter is configured to detect both edges of the input signal.

Note that the last two edges are not counted since the timer automatically clears the T_nEN bit after the current count matches the value in the **GPTMTnMR** register.

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



10.2.3.3 16-Bit Input Edge Time Mode

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

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

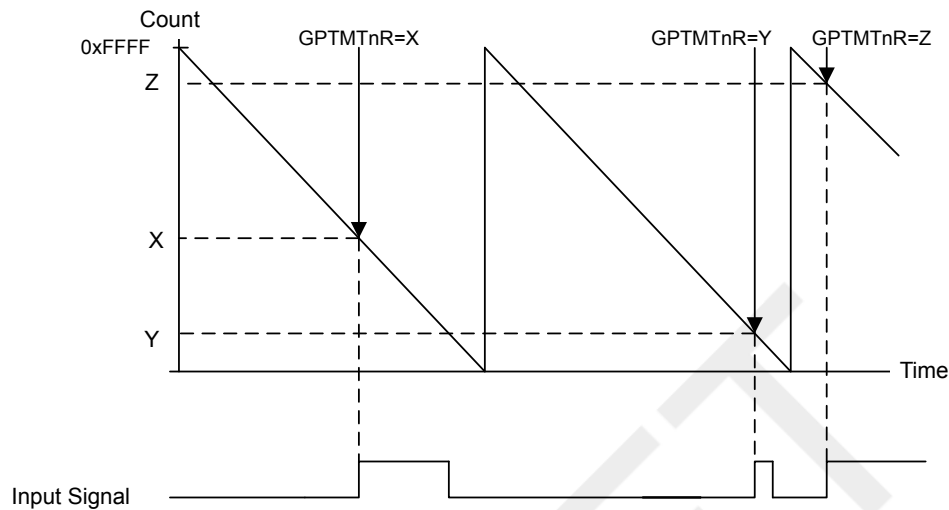
When software writes the **TnEN** bit in the **GPTMCTL** register, the timer is enabled for event capture. When the selected input event is detected, the current **Tn** counter value is captured in the **GPTMTnR** register and is available to be read by the controller. The GPTM then asserts the **CnERIS**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

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

Figure 10-3 on page 202 shows how input edge timing mode works. In the diagram, it is assumed that the start value of the timer is the default value of 0xFFFF, and the timer is configured to capture rising edge events.

Each time a rising edge event is detected, the current count value is loaded into the **GPTMTnR** register, and is held there until another rising edge is detected (at which point the new count value is loaded into **GPTMTnR**).

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



10.2.3.4 16-Bit PWM Mode

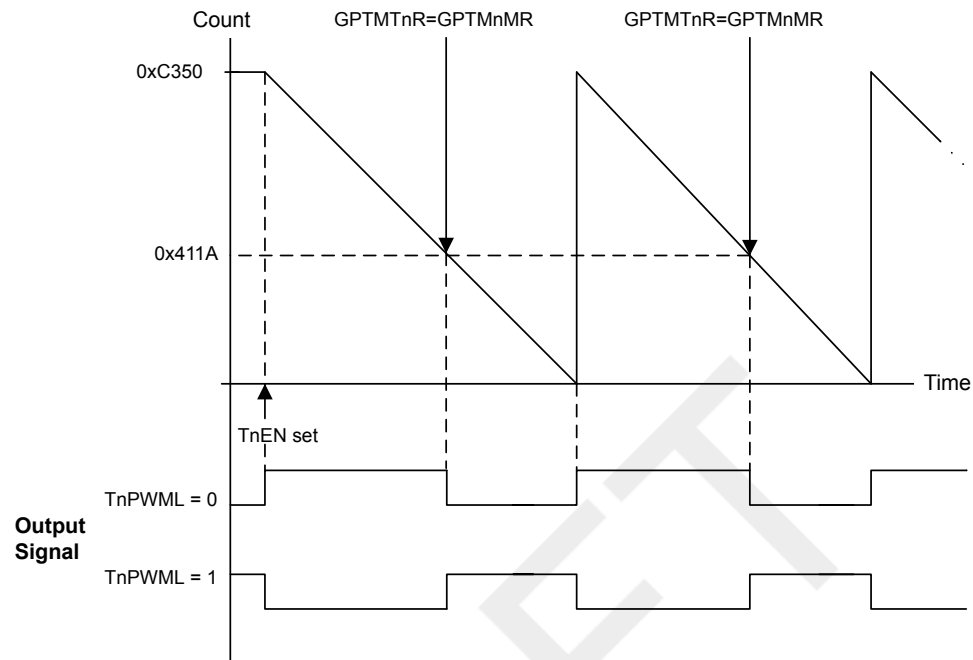
The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a down-counter with a start value (and thus period) defined by **GPTMTnILR**. PWM mode is enabled with the **GPTMTnMR** register by setting the T_nAMS bit to 0x1, the T_nCMR bit to 0x0, and the T_nMR field to 0x2.

When software writes the T_nEN bit in the **GPTMCTL** register, the counter begins counting down until it reaches the 0x0000 state. On the next counter cycle, the counter reloads its start value from **GPTMTnILR** (and **GPTMTnPR** if using a prescaler) and continues counting until disabled by software clearing the T_nEN bit in the **GPTMCTL** register. No interrupts or status bits are asserted in PWM mode.

The output PWM signal asserts when the counter is at the value of the **GPTMTnILR** register (its start state), and is deasserted when the counter value equals the value in the **GPTM Timern Match Register (GPTMnMATCHR)**. Software has the capability of inverting the output PWM signal by setting the T_nPWML bit in the **GPTMCTL** register.

Figure 10-4 on page 203 shows how to generate an output PWM with a 1-ms period and a 66% duty cycle assuming a 50-MHz input clock and $T_nPWML = 0$ (duty cycle would be 33% for the $T_nPWML = 1$ configuration). For this example, the start value is **GPTMnIRL**=0xC350 and the match value is **GPTMnMR**=0x411A.

Figure 10-4. 16-Bit PWM Mode Example



10.3 Initialization and Configuration

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

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

10.3.1 32-Bit One-Shot/Periodic Timer Mode

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

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

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

In One-Shot mode, the timer stops counting after 7 on page 204. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode does not stop counting after it times out.

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

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

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

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

10.3.3 16-Bit One-Shot/Periodic Timer Mode

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

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

In One-Shot mode, the timer stops counting after 8 on page 204. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode does not stop counting after it times out.

10.3.4 16-Bit Input Edge Count Mode

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

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

In Input Edge Count Mode, the timer stops after the desired number of edge events has been detected. To re-enable the timer, ensure that the $TnEN$ bit is cleared and repeat steps 4 on page 205-9 on page 205.

10.3.5 16-Bit Input Edge Timing Mode

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

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

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

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

10.3.6 16-Bit PWM Mode

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

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

In PWM Timing mode, the timer continues running after the PWM signal has been generated. The PWM period can be adjusted at any time by writing the **GPTMTnILR** register, and the change takes effect at the next cycle after the write.

10.4 Register Map

Table 10-2 on page 206 lists the GPTM registers. The offset listed is a hexadecimal increment to the register's address, relative to that timer's base address:

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

Table 10-2. Timers Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPTMCFG	R/W	0x0x0000.0000	GPTM Configuration	208
0x004	GPTMTAMR	R/W	0x0x0000.0000	GPTM TimerA Mode	209
0x008	GPTMTBMR	R/W	0x0x0000.0000	GPTM TimerB Mode	210
0x00C	GPTMCTL	R/W	0x0x0000.0000	GPTM Control	211

Offset	Name	Type	Reset	Description	See page
0x018	GPTMIMR	R/W	0x0x0000.0000	GPTM Interrupt Mask	213
0x01C	GPTMRIS	RO	0x0x0000.0000	GPTM Raw Interrupt Status	215
0x020	GPTMMIS	RO	0x0x0000.0000	GPTM Masked Interrupt Status	216
0x024	GPTMICR	W1C	0x0x0000.0000	GPTM Interrupt Clear	217
0x028	GPTMTAILR	R/W	0x0000.FFFF (16-bit mode) 0xFFFF.FFFF (32-bit mode)	GPTM TimerA Interval Load	219
0x02C	GPTMTBILR	R/W	0x0000.FFFF	GPTM TimerB Interval Load	220
0x030	GPTMTAMATCHR	R/W	0x0000.FFFF (16-bit mode) 0xFFFF.FFFF (32-bit mode)	GPTM TimerA Match	221
0x034	GPTMTBMATCHR	R/W	0x0000.FFFF	GPTM TimerB Match	222
0x038	GPTMTAPR	R/W	0x0000.0000	GPTM TimerA Prescale	223
0x03C	GPTMTBPR	R/W	0x0000.0000	GPTM TimerB Prescale	224
0x040	GPTMTAPMR	R/W	0x0000.0000	GPTM TimerA Prescale Match	225
0x044	GPTMTBPMR	R/W	0x0000.0000	GPTM TimerB Prescale Match	226
0x048	GPTMTAR	RO	0x0000.FFFF (16-bit mode) 0xFFFF.FFFF (32-bit mode)	GPTM TimerA	227
0x04C	GPTMTBR	RO	0x0000.FFFF	GPTM TimerB	228

10.5 Register Descriptions

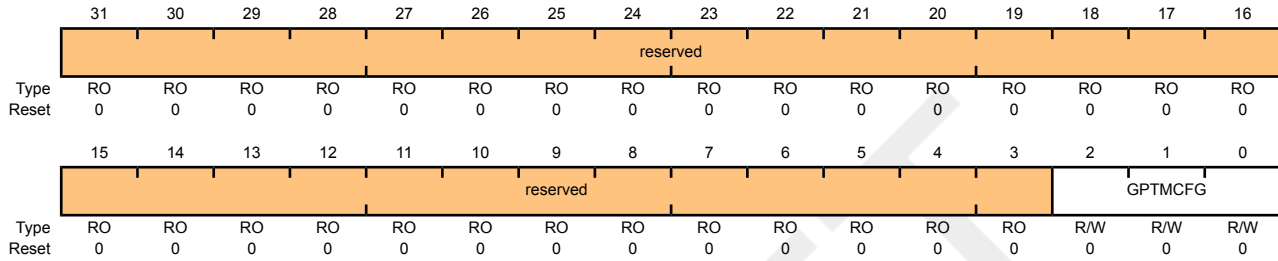
The remainder of this section lists and describes the GPTM registers, in numerical order by address offset.

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

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

GPTM Configuration (GPTMCFG)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x000
 Type R/W, reset 0x0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	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	GPTMCFG	R/W	0	GPTM Configuration 0x0: 32-bit timer configuration. 0x1: 32-bit real-time clock (RTC) counter configuration. 0x2: Reserved. 0x3: Reserved. 0x4-0x7: 16-bit timer configuration, function is controlled by bits 1:0 of GPTMTAMR and GPTMTBMR .

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

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

GPTM TimerA Mode (GPTMTAMR)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x004
 Type R/W, reset 0x0x0000.0000

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

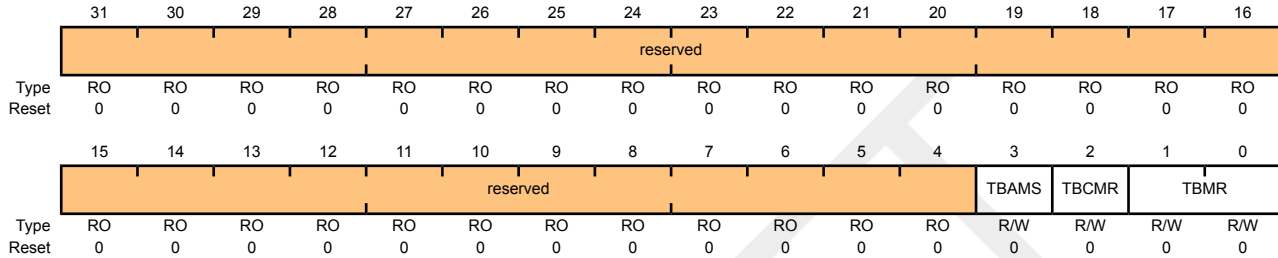
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TAAMS	R/W	0	GPTM TimerA Alternate Mode Select 0: Capture mode is enabled. 1: PWM mode is enabled. Note: To enable PWM mode, you must also clear the TACMR bit and set the TAMR field to 0x2.
2	TACMR	R/W	0	GPTM TimerA Capture Mode 0: Edge-Count mode. 1: Edge-Time mode.
1:0	TAMR	R/W	0	GPTM TimerA Mode 0x0: Reserved. 0x1: One-Shot Timer mode. 0x2: Periodic Timer mode. 0x3: Capture mode. The Timer mode is based on the timer configuration defined by bits 2:0 in the GPTMCFG register (16-or 32-bit). In 16-bit timer configuration, TAMR controls the 16-bit timer modes for TimerA. In 32-bit timer configuration, this register controls the mode and the contents of GPTMTBMR are ignored.

Register 3: GPTM TimerB Mode (GPTMTBMR), offset 0x008

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

GPTM TimerB Mode (GPTMTBMR)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x008
 Type R/W, reset 0x0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TBAMS	R/W	0	GPTM TimerB Alternate Mode Select 0: Capture mode is enabled. 1: PWM mode is enabled. Note: To enable PWM mode, you must also clear the TBCMR bit and set the TBMR field to 0x2.
2	TBCMR	R/W	0	GPTM TimerB Capture Mode 0: Edge-Count mode. 1: Edge-Time mode.
1:0	TBMR	R/W	0	GPTM TimerB Mode 0x0: Reserved. 0x1: One-Shot Timer mode. 0x2: Periodic Timer mode. 0x3: Capture mode. The timer mode is based on the timer configuration defined by bits 2:0 in the GPTMCFG register. In 16-bit timer configuration, these bits control the 16-bit timer modes for TimerB. In 32-bit timer configuration, this register's contents are ignored and GPTMTAMR is used.

Register 4: GPTM Control (GPTMCTL), offset 0x00C

This register is used alongside the **GPTMCFG** and **GMTMTnMR** registers to fine-tune the timer configuration, and to enable other features such as timer stall and the output trigger. The output trigger can be used to initiate transfers on the ADC module.

GPTM Control (GPTMCTL)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x00C
 Type R/W, reset 0x0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	TBPWML	TBOTE	reserved	TBEVENT	TBSTALL	TBEN	reserved	TAPWML	TAOTE	RTCEN	TAEVENT	TASTALL	TAEN		
Type	RO	R/W	R/W	RO	R/W	R/W	R/W	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	TBPWML	R/W	0	GPTM TimerB PWM Output Level 0: Output is unaffected. 1: Output is inverted.
13	TBOTE	R/W	0	GPTM TimerB Output Trigger Enable 0: The output TimerB trigger is disabled. 1: The output TimerB trigger is enabled.
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11:10	TBEVENT	R/W	0	GPTM TimerB Event Mode 00: Positive edge. 01: Negative edge. 10: Reserved. 11: Both edges.
9	TBSTALL	R/W	0	GPTM TimerB Stall Enable 0: TimerB stalling is disabled. 1: TimerB stalling is enabled.

Bit/Field	Name	Type	Reset	Description
8	TBEN	R/W	0	GPTM TimerB Enable 0: TimerB is disabled. 1: TimerB 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 TimerA PWM Output Level 0: Output is unaffected. 1: Output is inverted.
5	TAOTE	R/W	0	GPTM TimerA Output Trigger Enable 0: The output TimerA trigger is disabled. 1: The output TimerA trigger is enabled.
4	RTCEN	R/W	0	GPTM RTC Enable 0: RTC counting is disabled. 1: RTC counting is enabled.
3:2	TAEVENT	R/W	0	GPTM TimerA Event Mode 00: Positive edge. 01: Negative edge. 10: Reserved. 11: Both edges.
1	TASTALL	R/W	0	GPTM TimerA Stall Enable 0: TimerA stalling is disabled. 1: TimerA stalling is enabled.
0	TAEN	R/W	0	GPTM TimerA Enable 0: TimerA is disabled. 1: TimerA is enabled and begins counting or the capture logic is enabled based on the GPTMCFG register.

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

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

GPTM Interrupt Mask (GPTMIMR)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x018
 Type R/W, reset 0x0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
1	CAMIM	R/W	0	GPTM CaptureA Match Interrupt Mask 0: Interrupt is disabled. 1: Interrupt is enabled.
0	TATOIM	R/W	0	GPTM TimerA Time-Out Interrupt Mask 0: Interrupt is disabled. 1: Interrupt is enabled.

DRAFT

Register 6: GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C

This register shows the state of the GPTM's internal interrupt signal. These bits are set whether or not the interrupt is masked in the **GPTMIMR** register. Each bit can be cleared by writing a 1 to its corresponding bit in **GPTMICR**.

GPTM Raw Interrupt Status (GPTMRIS)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x01C
 Type RO, reset 0x0x0000.0000

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

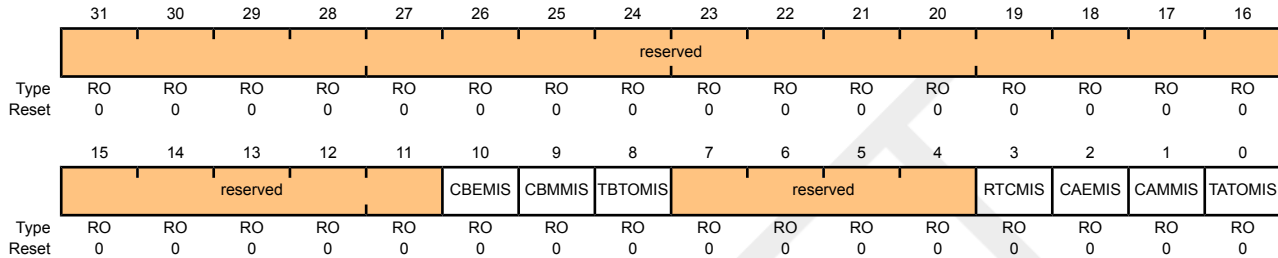
Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBERIS	RO	0	GPTM CaptureB Event Raw Interrupt This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

Register 7: GPTM Masked Interrupt Status (GPTMMIS), offset 0x020

This register show the state of the GPTM's controller-level interrupt. If an interrupt is unmasked in **GPTMIMR**, and there is an event that causes the interrupt to be asserted, the corresponding bit is set in this register. All bits are cleared by writing a 1 to the corresponding bit in **GPTMICR**.

GPTM Masked Interrupt Status (GPTMMIS)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x020
 Type RO, reset 0x0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:11	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	CBEMIS	RO	0	GPTM CaptureB Event Masked Interrupt This is the CaptureB event interrupt status after masking.
9	CBMMIS	RO	0	GPTM CaptureB Match Masked Interrupt This is the CaptureB match interrupt status after masking.
8	TBTOMIS	RO	0	GPTM TimerB Time-Out Masked Interrupt This is the TimerB time-out interrupt status after masking.
7:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	RTCMIS	RO	0	GPTM RTC Masked Interrupt This is the RTC event interrupt status after masking.
2	CAEMIS	RO	0	GPTM CaptureA Event Masked Interrupt This is the CaptureA event interrupt status after masking.
1	CAMMIS	RO	0	GPTM CaptureA Match Masked Interrupt This is the CaptureA match interrupt status after masking.
0	TATOMIS	RO	0	GPTM TimerA Time-Out Masked Interrupt This is the TimerA time-out interrupt status after masking.

Register 8: GPTM Interrupt Clear (GPTMICR), offset 0x024

This register is used to clear the status bits in the **GPTMRIS** and **GPTMMIS** registers. Writing a 1 to a bit clears the corresponding bit in the **GPTMRIS** and **GPTMMIS** registers.

GPTM Interrupt Clear (GPTMICR)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x024
 Type W1C, reset 0x0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBECINT	W1C	0	GPTM CaptureB Event Interrupt Clear 0: The interrupt is unaffected. 1: The interrupt is cleared.
9	CBMCINT	W1C	0	GPTM CaptureB Match Interrupt Clear 0: The interrupt is unaffected. 1: The interrupt is cleared.
8	TBTOCINT	W1C	0	GPTM TimerB Time-Out Interrupt Clear 0: The interrupt is unaffected. 1: The interrupt is cleared.
7:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	RTCCINT	W1C	0	GPTM RTC Interrupt Clear 0: The interrupt is unaffected. 1: The interrupt is cleared.
2	CAECINT	W1C	0	GPTM CaptureA Event Interrupt Clear 0: The interrupt is unaffected. 1: The interrupt is cleared.
1	CAMCINT	W1C	0	GPTM CaptureA Match Raw Interrupt This is the CaptureA match interrupt status after masking.

Bit/Field	Name	Type	Reset	Description
0	TATOCINT	W1C	0	GPTM TimerA Time-Out Raw Interrupt 0: The interrupt is unaffected. 1: The interrupt is cleared.

DRAFT

Register 9: GPTM TimerA Interval Load (GPTMTAILR), offset 0x028

This register is used to load the starting count value into the timer. When GPTM is configured to one of the 32-bit modes, **GPTMTAILR** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM TimerB Interval Load (GPTMTBILR)** register). In 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of **GPTMTBILR**.

GPTM TimerA Interval Load (GPTMTAILR)

Timer0 base: 0x4003.0000

Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x028

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

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

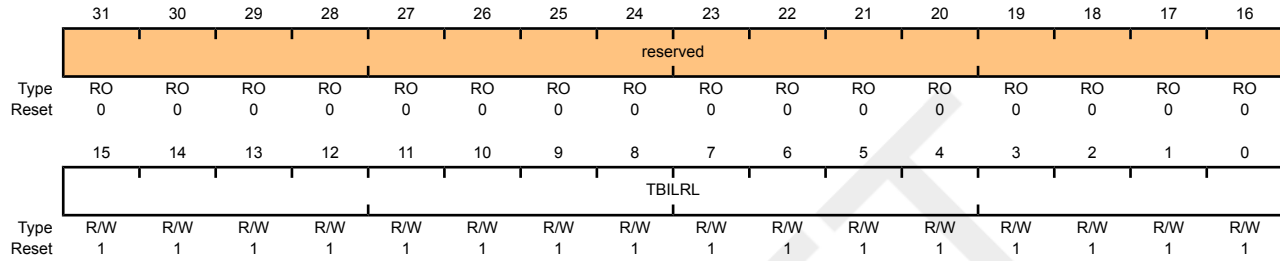
Bit/Field	Name	Type	Reset	Description
31:16	TAILRH	R/W	0xFFFF (32-bit mode) 0x0000 (16-bit mode)	GPTM TimerA Interval Load Register High When configured for 32-bit mode via the GPTMCFG register, the GPTM TimerB Interval Load (GPTMTBILR) register loads this value on a write. A read returns the current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

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

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

GPTM TimerB Interval Load (GPTMTBILR)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x02C
 Type R/W, reset 0x0000.FFFF



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TBILRL	R/W	0xFFFF	GPTM TimerB Interval Load Register When the GPTM is not configured as a 32-bit timer, a write to this field updates GPTMTBILR . In 32-bit mode, writes are ignored, and reads return the current value of GPTMTBILR .

Register 11: GPTM TimerA Match (GPTMTAMATCHR), offset 0x030

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

GPTM TimerA Match (GPTMTAMATCHR)

Timer0 base: 0x4003.0000

Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x030

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

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

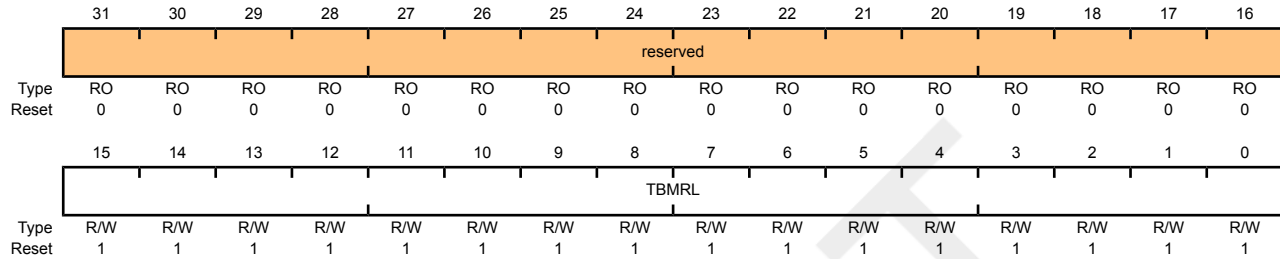
Bit/Field	Name	Type	Reset	Description
31:16	TAMRH	R/W	0xFFFF (32-bit mode) 0x0000 (16-bit mode)	<p>GPTM TimerA Match Register High</p> <p>When configured for 32-bit Real-Time Clock (RTC) mode via the GPTMCFG register, this value is compared to the upper half of GPTMTAR, to determine match events.</p>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

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

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

GPTM TimerB Match (GPTMTBMATCHR)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x034
 Type R/W, reset 0x0000.FFFF



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

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

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

GPTM TimerA Prescale (GPTMTAPR)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x038
 Type R/W, reset 0x0000.0000

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

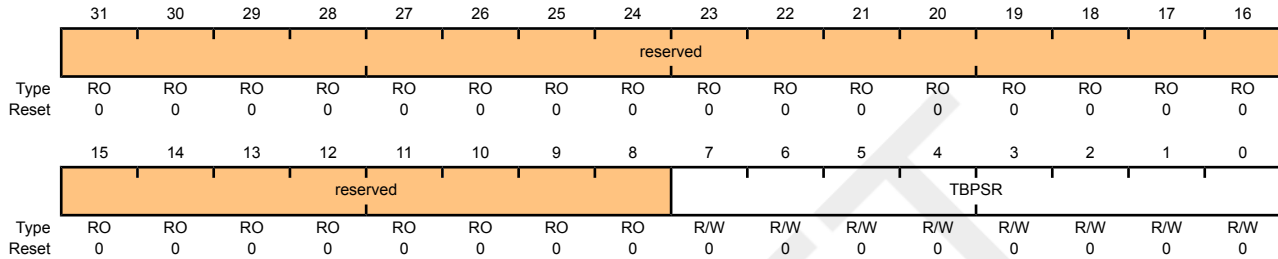
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TAPSR	R/W	0	GPTM TimerA Prescale The register loads this value on a write. A read returns the current value of the register. Refer to Table 10-1 on page 200 for more details and an example.

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

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

GPTM TimerB Prescale (GPTMTBPR)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x03C
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TBPSR	R/W	0	GPTM TimerB Prescale The register loads this value on a write. A read returns the current value of this register. Refer to Table 10-1 on page 200 for more details and an example.

Register 15: GPTM TimerA Prescale Match (GPTMTAPMR), offset 0x040

This register effectively extends the range of **GPTMTAMATCHR** to 24 bits when operating in 16-bit one-shot or periodic mode.

GPTM TimerA Prescale Match (GPTMTAPMR)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x040
 Type R/W, reset 0x0000.0000

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

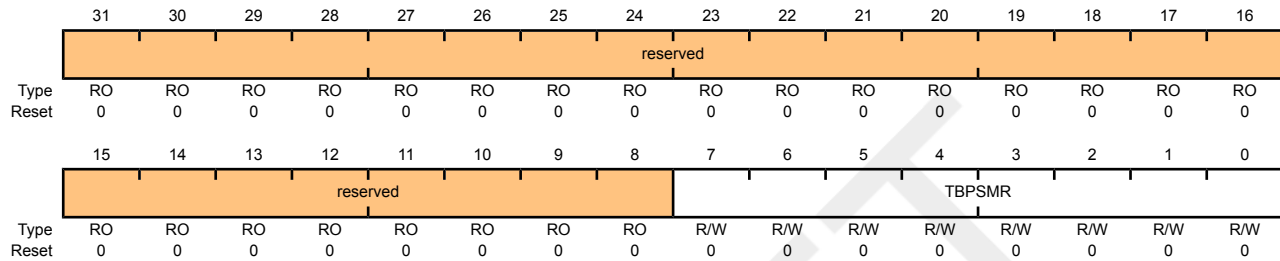
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TAPSMR	R/W	0	GPTM TimerA Prescale Match This value 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 **GPTMTBMATCHR** to 24 bits when operating in 16-bit one-shot or periodic mode.

GPTM TimerB Prescale Match (GPTMTBPMR)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x044
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved 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	0	GPTM TimerB Prescale Match This value is used alongside GPTMTBMATCHR to detect timer match events while using a prescaler.

Register 17: GPTM TimerA (GPTMTAR), offset 0x048

This register shows the current value of the TimerA counter in all cases except for Input Edge Count mode. When in this mode, this register contains the time at which the last edge event took place.

GPTM TimerA (GPTMTAR)

Timer0 base: 0x4003.0000

Timer1 base: 0x4003.1000

Timer2 base: 0x4003.2000

Timer3 base: 0x4003.3000

Offset 0x048

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

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

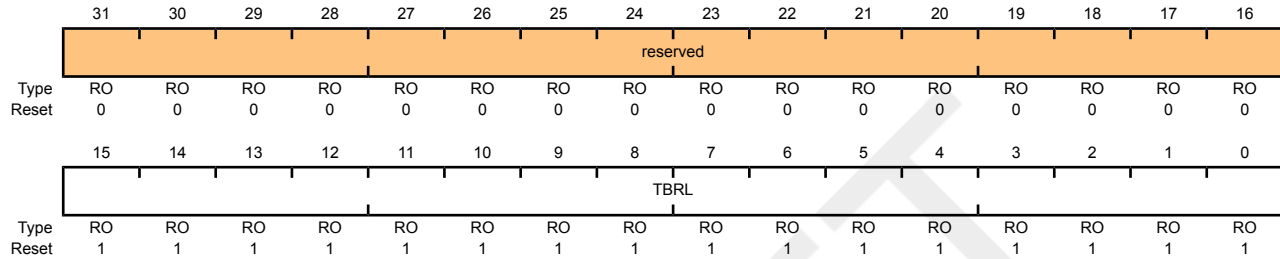
Bit/Field	Name	Type	Reset	Description
31:16	TARH	RO	0xFFFF (32-bit mode) 0x0000 (16-bit mode)	GPTM TimerA Register High If the GPTMCFG is in a 32-bit mode, TimerB value is read. If the GPTMCFG is in a 16-bit mode, this is read as zero.
15:0	TARL	RO	0xFFFF	GPTM TimerA Register Low A read returns the current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

Register 18: GPTM TimerB (GPTMTBR), offset 0x04C

This register shows the current value of the TimerB counter in all cases except for Input Edge Count mode. When in this mode, this register contains the time at which the last edge event took place.

GPTM TimerB (GPTMTBR)

Timer0 base: 0x4003.0000
 Timer1 base: 0x4003.1000
 Timer2 base: 0x4003.2000
 Timer3 base: 0x4003.3000
 Offset 0x04C
 Type RO, reset 0x0000.FFFF



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TBRL	RO	0xFFFF	GPTM TimerB A read returns the current value of the GPTM TimerB Count Register , except in Input Edge Count mode, when it returns the timestamp from the last edge event.

11 Watchdog Timer

WDT

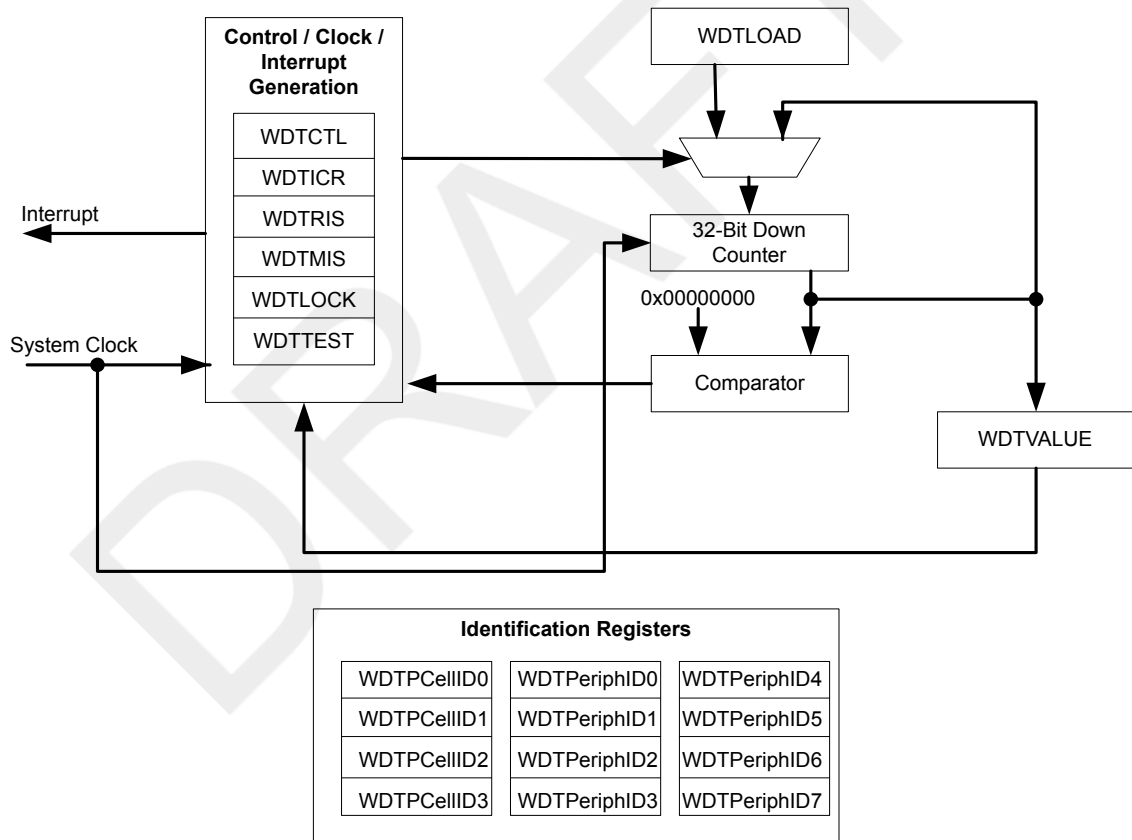
A watchdog timer can generate nonmaskable interrupts (NMIs) or a reset when a time-out value is reached. The watchdog timer is used to regain control when a system has failed due to a software error or due to the failure of an external device to respond in the expected way.

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

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

11.1 Block Diagram

Figure 11-1. WDT Module Block Diagram



11.2 Functional Description

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

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.

If the timer counts down to its zero state again before the first time-out interrupt is cleared, and the reset signal has been enabled (via the `WatchdogResetEnable` function), the Watchdog timer asserts its reset signal to the system. If the interrupt is cleared before the 32-bit counter reaches its second time-out, the 32-bit counter is loaded with the value in the **WDTLOAD** register, and counting resumes from that value.

If **WDTLOAD** is written with a new value while the Watchdog Timer counter is counting, then the counter is loaded with the new value and continues counting.

Writing to **WDTLOAD** does not clear an active interrupt. An interrupt must be specifically cleared by writing to the **Watchdog Interrupt Clear (WDTICR)** register.

The Watchdog module interrupt and reset generation can be enabled or disabled as required. When the interrupt is re-enabled, the 32-bit counter is preloaded with the load register value and not its last state.

11.3 Initialization and Configuration

To use the WDT, its peripheral clock must be enabled by setting the `WDT` bit in the **RCGC0** register. The Watchdog Timer is configured using the following sequence:

1. Load the **WDTLOAD** register with the desired timer load value.
2. If the Watchdog is configured to trigger system resets, set the `RESEN` bit in the **WDTCTL** register.
3. Set the `INTEN` bit in the **WDTCTL** register to enable the Watchdog and lock the control register.

If software requires that all of the watchdog registers are locked, the Watchdog Timer module can be fully locked by writing any value to the **WDTLOCK** register. To unlock the Watchdog Timer, write a value of `0x1ACCE551`.

11.4 Register Map

Table 11-1 on page 230 lists the Watchdog registers. The offset listed is a hexadecimal increment to the register's address, relative to the Watchdog Timer base address of `0x4000.0000`.

Table 11-1. Watchdog Timer Register Map

Offset	Name	Type	Reset	Description	See page
0x000	WDTLOAD	R/W	0xFFFF.FFFF	Watchdog Load	232
0x004	WDTVALUE	RO	0xFFFF.FFFF	Watchdog Value	233
0x008	WDTCTL	R/W	0x0000.0000	Watchdog Control	234
0x00C	WDTICR	WO	-	Watchdog Interrupt Clear	235
0x010	WDTRIS	RO	0x0000.0000	Watchdog Raw Interrupt Status	236

Offset	Name	Type	Reset	Description	See page
0x014	WDTMIS	RO	0x0000.0000	Watchdog Masked Interrupt Status	237
0x418	WDTTEST	R/W	0x0000.0000	Watchdog Test	238
0xC00	WDTLOCK	R/W	0x0000.0000	Watchdog Lock	239
0xFD0	WDTPeriphID4	RO	0x0000.0000	Watchdog Peripheral Identification 4	240
0xFD4	WDTPeriphID5	RO	0x0000.0000	Watchdog Peripheral Identification 5	241
0xFD8	WDTPeriphID6	RO	0x0000.0000	Watchdog Peripheral Identification 6	242
0xFDC	WDTPeriphID7	RO	0x0000.0000	Watchdog Peripheral Identification 7	243
0xFE0	WDTPeriphID0	RO	0x0000.0005	Watchdog Peripheral Identification 0	244
0xFE4	WDTPeriphID1	RO	0x0000.0018	Watchdog Peripheral Identification 1	245
0xFE8	WDTPeriphID2	RO	0x0000.0018	Watchdog Peripheral Identification 2	246
0xFEC	WDTPeriphID3	RO	0x0000.0001	Watchdog Peripheral Identification 3	247
0xFF0	WDTPrimeCellID0	RO	0x0000.000D	Watchdog PrimeCell Identification 0	248
0xFF4	WDTPrimeCellID1	RO	0x0000.00F0	Watchdog PrimeCell Identification 1	249
0xFF8	WDTPrimeCellID2	RO	0x0000.0005	Watchdog PrimeCell Identification 2	250
0xFFC	WDTPrimeCellID3	RO	0x0000.00B1	Watchdog PrimeCell Identification 3	251

11.5 Register Descriptions

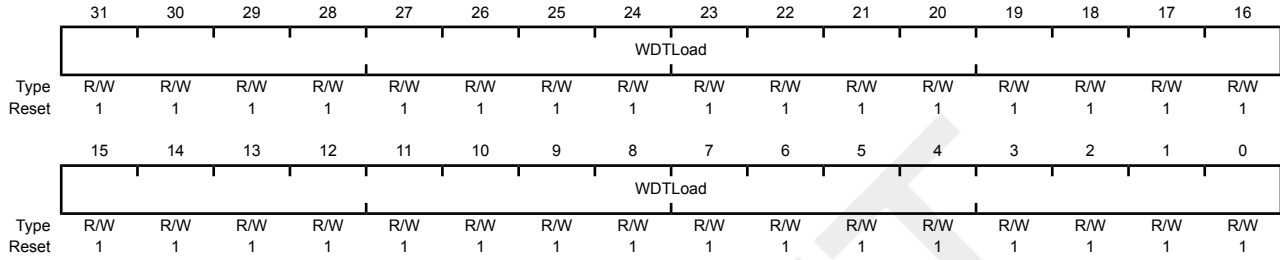
The remainder of this section lists and describes the WDT registers, in numerical order by address offset.

Register 1: Watchdog Load (WDTLOAD), offset 0x000

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

Watchdog Load (WDTLOAD)

Base 0x4000.0000
 Offset 0x000
 Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	WDTLoad	R/W	0xFFFF.FFFF	Watchdog Load Value

Register 2: Watchdog Value (WDTVALUE), offset 0x004

This register contains the current count value of the timer.

Watchdog Value (WDTVALUE)

Base 0x4000.0000

Offset 0x004

Type RO, reset 0xFFFF.FFFF

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

Bit/Field	Name	Type	Reset	Description
31:0	WDTValue	RO	0xFFFF.FFFF	Watchdog Value Current value of the 32-bit down counter.

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

This register is the watchdog control register. The watchdog timer can be configured to generate a reset signal (on second time-out) or an interrupt on time-out.

When the watchdog interrupt has been enabled, all subsequent writes to the control register are ignored. The only mechanism that can re-enable writes is a hardware reset.

Watchdog Control (WDTCTL)

Base 0x4000.0000
 Offset 0x008
 Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of 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 0: Disabled. 1: Enable the Watchdog module reset output.
0	INTEN	R/W	0	Watchdog Interrupt Enable 0: Interrupt event disabled (once this bit is set, it can only be cleared by a hardware reset). 1: Interrupt event enabled. Once enabled, all writes are ignored.

Register 4: Watchdog Interrupt Clear (WDTICR), offset 0x00C

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

Watchdog Interrupt Clear (WDTICR)

Base 0x4000.0000

Offset 0x00C

Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WDTIntClr															
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
	WDTIntClr															
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	WDTIntClr	WO	-	Watchdog Interrupt Clear

Register 5: Watchdog Raw Interrupt Status (WDTRIS), offset 0x010

This register is the raw interrupt status register. Watchdog interrupt events can be monitored via this register if the controller interrupt is masked.

Watchdog Raw Interrupt Status (WDTRIS)

Base 0x4000.0000
 Offset 0x010
 Type RO, reset 0x0000.0000

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

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

Register 6: Watchdog Masked Interrupt Status (WDTMIS), offset 0x014

This register is the masked interrupt status register. The value of this register is the logical AND of the raw interrupt bit and the Watchdog interrupt enable bit.

Watchdog Masked Interrupt Status (WDTMIS)

Base 0x4000.0000

Offset 0x014

Type RO, reset 0x0000.0000

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

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

Register 7: Watchdog Test (WDTTEST), offset 0x418

This register provides user-enabled stalling when the microcontroller asserts the CPU halt flag during debug.

Watchdog Test (WDTTEST)

Base 0x4000.0000
 Offset 0x418
 Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	STALL	R/W	0	Watchdog Stall Enable When set to 1, if the Stellaris® microcontroller is stopped with a debugger, the watchdog timer stops counting. Once the microcontroller is restarted, the watchdog timer resumes counting.
7:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 8: Watchdog Lock (WDTLOCK), offset 0xC00

Writing 0x1ACCE551 to the **WDTLOCK** register enables write access to all other registers. Writing any other value to the **WDTLOCK** register re-enables the locked state for register writes to all the other registers. Reading the **WDTLOCK** register returns the lock status rather than the 32-bit value written. Therefore, when write accesses are disabled, reading the **WDTLOCK** register returns 0x0000.0001 (when locked; otherwise, the returned value is 0x0000.0000 (unlocked)).

Watchdog Lock (WDTLOCK)

Base 0x4000.0000
Offset 0xC00
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WDTLock															
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
	WDTLock															
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

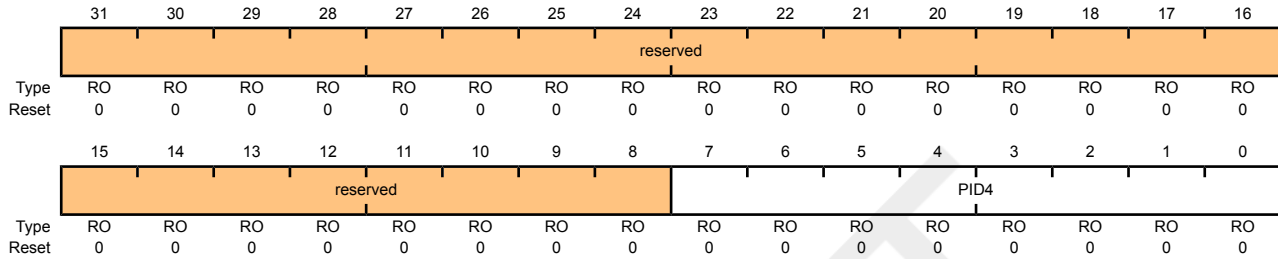
Bit/Field	Name	Type	Reset	Description
31:0	WDTLock	R/W	0x0000	<p>Watchdog Lock</p> <p>A write of the value 0x1ACCE551 unlocks the watchdog registers for write access. A write of any other value reapplies the lock, preventing any register updates.</p> <p>A read of this register returns the following values:</p> <p>Locked: 0x0000.0001</p> <p>Unlocked: 0x0000.0000</p>

Register 9: Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0

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

Watchdog Peripheral Identification 4 (WDTPeriphID4)

Base 0x4000.0000
 Offset 0xFD0
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	WDT Peripheral ID Register[7:0]

Register 10: Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4

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

Watchdog Peripheral Identification 5 (WDTPeriphID5)

Base 0x4000.0000

Offset 0xFD4

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID5							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	WDT Peripheral ID Register[15:8]

Register 11: Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8

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

Watchdog Peripheral Identification 6 (WDTPeriphID6)

Base 0x4000.0000
 Offset 0xFD8
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID6							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Register 12: Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC

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

Watchdog Peripheral Identification 7 (WDTPeriphID7)

Base 0x4000.0000

Offset 0xFDC

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID7							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	WDT Peripheral ID Register[31:24]

Register 13: Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0

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

Watchdog Peripheral Identification 0 (WDTPeriphID0)

Base 0x4000.0000
 Offset 0xFE0
 Type RO, reset 0x0000.0005

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

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

Register 14: Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4

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

Watchdog Peripheral Identification 1 (WDTPeriphID1)

Base 0x4000.0000

Offset 0xFE4

Type RO, reset 0x0000.0018

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x18	Watchdog Peripheral ID Register[15:8]

Register 15: Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8

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

Watchdog Peripheral Identification 2 (WDTPeriphID2)

Base 0x4000.0000
 Offset 0xFE8
 Type RO, reset 0x0000.0018

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	Watchdog Peripheral ID Register[23:16]

Register 16: Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC

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

Watchdog Peripheral Identification 3 (WDTPeriphID3)

Base 0x4000.0000

Offset 0xFEC

Type RO, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

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

Register 17: Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0

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

Watchdog PrimeCell Identification 0 (WDTPCellID0)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

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

Register 18: Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4

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

Watchdog PrimeCell Identification 1 (WDTPCellID1)

Base 0x4000.0000

Offset 0xFF4

Type RO, reset 0x0000.00F0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

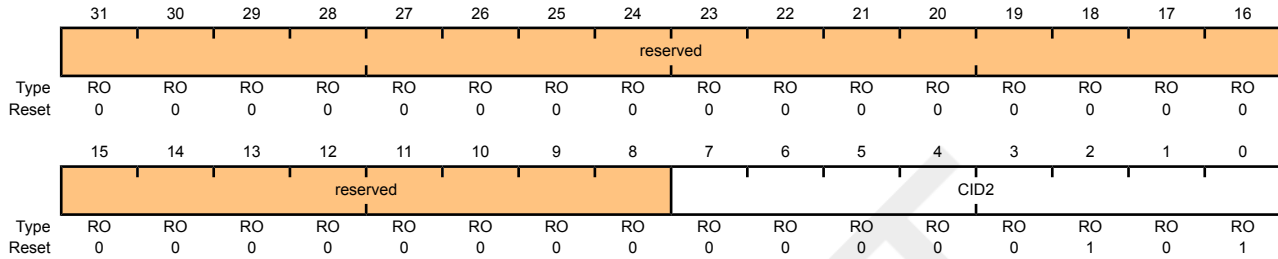
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	Watchdog PrimeCell ID Register[15:8]

Register 19: Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8

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

Watchdog PrimeCell Identification 2 (WDTPCellID2)

Base 0x4000.0000
 Offset 0xFF8
 Type RO, reset 0x0000.0005



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

Register 20: Watchdog PrimeCell Identification 3 (WDTPCellID3), offset 0xFFC

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

Watchdog PrimeCell Identification 3 (WDTPCellID3)

Base 0x4000.0000

Offset 0xFFC

Type RO, reset 0x0000.00B1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved 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)

ADC

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

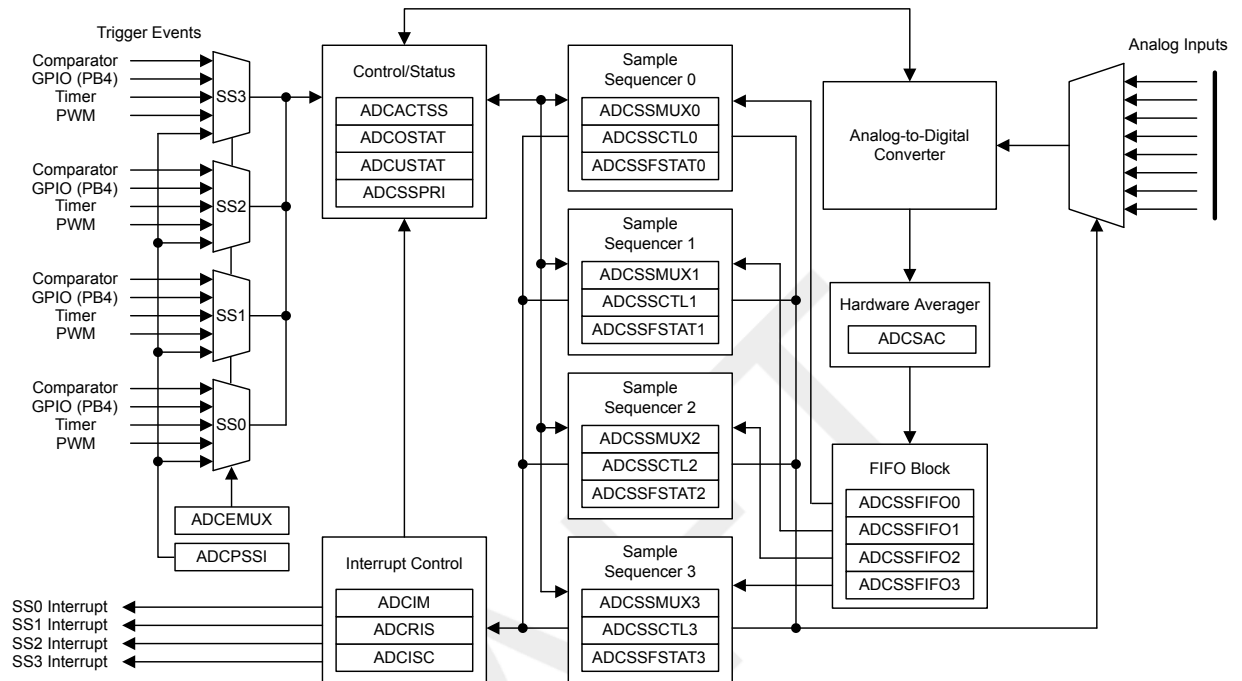
The Stellaris[®] ADC module features 10-bit conversion resolution and supports eight input channels, plus an internal temperature sensor. The ADC module contains a programmable sequencer which allows for the sampling of multiple analog input sources without controller intervention. Each sample sequence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

The Stellaris[®] ADC provides the following features:

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

12.1 Block Diagram

Figure 12-1. ADC Module Block Diagram



12.2 Functional Description

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

12.2.1 Sample Sequencers

The sampling control and data capture is handled by the Sample Sequencers. All of the sequencers are identical in implementation except for the number of samples that can be captured and the depth of the FIFO. Table 12-1 on page 253 shows the maximum number of samples that each Sequencer can capture and its corresponding FIFO depth. In this implementation, each FIFO entry is a 32-bit word, with the lower 10 bits containing the conversion result.

Table 12-1. Samples and FIFO Depth of Sequencers

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

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

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

After a sample sequence completes execution, the result data can be retrieved from the **ADC Sample Sequence Result FIFO (ADCSSFIFO)** registers. The FIFOs are simple circular buffers that read a single address to "pop" result data. For software debug purposes, the positions of the FIFO head and tail pointers are visible in the **ADC Sample Sequence FIFO Status (ADCSSFSTATn)** registers along with **FULL** and **EMPTY** status flags. Overflow and underflow conditions are monitored using the **ADCSTAT** and **ADCUSTAT** registers.

12.2.2 Module Control

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

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

12.2.2.1 Interrupts

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

12.2.2.2 Prioritization

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

12.2.2.3 Sampling Events

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

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

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

12.2.3 Hardware Sample Averaging Circuit

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

By default the averaging circuit is off and all data from the converter passes through to the sequencer FIFO. The averaging hardware is controlled by the **ADC Sample Averaging Control (ADCSAC)** register (see page 268). There is a single averaging circuit and all input channels receive the same amount of averaging whether they are single-ended or differential.

12.2.4 Analog-to-Digital Converter

The converter itself generates a 10-bit output value for selected analog input. Special analog pads are used to minimize the distortion on the input.

12.2.5 Test Modes

There is a user-available test mode that allows for loopback operation within the digital portion of the ADC module. This can be useful for debugging software without having to provide actual analog stimulus. This mode is available through the **ADC Test Mode Loopback (ADCTMLB)** register (see page 283).

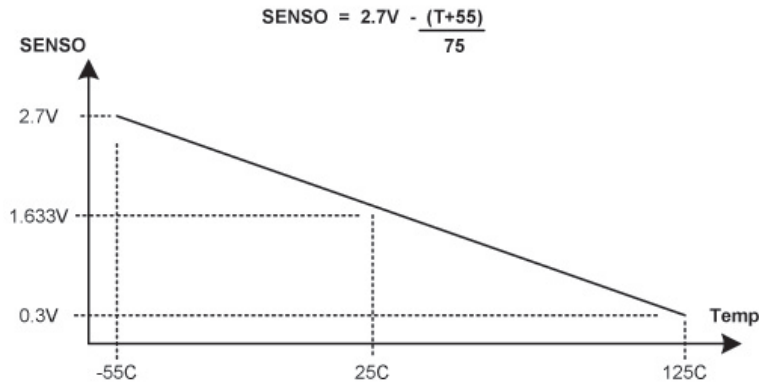
12.2.6 Internal Temperature Sensor

The internal temperature sensor provides an analog temperature reading as well as a reference voltage. The voltage at the output terminal $SENSO$ is given by the following equation:

$$SENSO = 2.7 - ((T + 55) / 75)$$

This relation is shown in Figure 12-2 on page 256.

Figure 12-2. Internal Temperature Sensor Characteristic



12.3 Initialization and Configuration

In order for the ADC module to be used, the PLL must be enabled and using a supported crystal frequency (see the **RCC** register). Using unsupported frequencies can cause faulty operation in the ADC module.

12.3.1 Module Initialization

Initialization of the ADC module is a simple process with very few steps. The main steps include enabling the clock to the ADC and reconfiguring the Sample Sequencer priorities (if needed).

The initialization sequence for the ADC is as follows:

1. Enable the ADC clock by writing a value of 0x0001.0000 to the **RCCG1** register (see page 97).
2. If required by the application, reconfigure the Sample Sequencer priorities in the **ADCSSPRI** register. The default configuration has Sample Sequencer 0 with the highest priority, and Sample Sequencer 3 as the lowest priority.

12.3.2 Sample Sequencer Configuration

Configuration of the Sample Sequencers is slightly more complex than the module initialization since each sample sequence is completely programmable.

The configuration for each Sample Sequencer should be as follows:

1. Ensure that the Sample Sequencer is disabled by writing a 0 to the corresponding **ASEN** bit in the **ADCACTSS** register. Programming of the Sample Sequencers is allowed without having them enabled. Disabling the Sequencer during programming prevents erroneous execution if a trigger event were to occur during the configuration process.
2. Configure the trigger event for the Sample Sequencer in the **ADCEMUX** register.
3. For each sample in the sample sequence, configure the corresponding input source in the **ADCSSMUXn** register.

4. For each sample in the sample sequence, configure the sample control bits in the corresponding nibble in the **ADCSSCTLn** register. When programming the last nibble, ensure that the **END** bit is set. Failure to set the **END** bit causes unpredictable behavior.
5. If interrupts are to be used, write a 1 to the corresponding **MASK** bit in the **ADCIM** register.
6. Enable the Sample Sequencer logic by writing a 1 to the corresponding **ASEN** bit in the **ADCACTSS** register.

12.4 Register Map

Table 12-2 on page 257 lists the ADC registers. The offset listed is a hexadecimal increment to the register's address, relative to the ADC base address of 0x4003.8000.

Table 12-2. ADC Register Map

Offset	Name	Type	Reset	Description	See page
0x000	ADCACTSS	R/W	0x0000.0000	ADC Active Sample Sequencer	259
0x004	ADCRIS	RO	0x0000.0000	ADC Raw Interrupt Status	260
0x008	ADCIM	R/W	0x0000.0000	ADC Interrupt Mask	261
0x00C	ADCISC	R/W1C	0x0000.0000	ADC Interrupt Status and Clear	262
0x010	ADCOSTAT	R/W1C	0x0000.0000	ADC Overflow Status	263
0x014	ADCEMUX	R/W	0x0000.0000	ADC Event Multiplexer Select	264
0x018	ADCUSTAT	R/W1C	0x0000.0000	ADC Underflow Status	265
0x020	ADCSSPRI	R/W	0x0000.3210	ADC Sample Sequencer Priority	266
0x028	ADCPSSI	WO	-	ADC Processor Sample Sequence Initiate	267
0x030	ADCSAC	R/W	0x0000.0000	ADC Sample Averaging Control	268
0x040	ADCSSMUX0	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 0	269
0x044	ADCSSCTL0	R/W	0x0000.0000	ADC Sample Sequence Control 0	271
0x048	ADCSSFIFO0	RO	0x0000.0000	ADC Sample Sequence Result FIFO 0	273
0x04C	ADCSSFSTAT0	RO	0x0000.0100	ADC Sample Sequence FIFO 0 Status	274
0x060	ADCSSMUX1	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 1	275
0x064	ADCSSCTL1	R/W	0x0000.0000	ADC Sample Sequence Control 1	276
0x068	ADCSSFIFO1	RO	0x0000.0000	ADC Sample Sequence Result FIFO 1	273
0x06C	ADCSSFSTAT1	RO	0x0000.0100	ADC Sample Sequence FIFO 1 Status	274
0x080	ADCSSMUX2	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 2	277
0x084	ADCSSCTL2	R/W	0x0000.0000	ADC Sample Sequence Control 2	278
0x088	ADCSSFIFO2	RO	0x0000.0000	ADC Sample Sequence Result FIFO 2	273
0x08C	ADCSSFSTAT2	RO	0x0000.0100	ADC Sample Sequence FIFO 2 Status	274
0x0A0	ADCSSMUX3	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 3	279

Offset	Name	Type	Reset	Description	See page
0x0A4	ADCSSCTL3	R/W	0x0000.0002	ADC Sample Sequence Control 3	280
0x0A8	ADCSSFIFO3	RO	0x0000.0000	ADC Sample Sequence Result FIFO 3	281
0x0AC	ADCSSFSTAT3	RO	0x0000.0100	ADC Sample Sequence FIFO 3 Status	282
0x100	ADCTMLB	R/W	0x0000.0000	ADC Test Mode Loopback	283

12.5 Register Descriptions

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

DRAFT

Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000

This register controls the activation of the Sample Sequencers. Each Sample Sequencer can be enabled/disabled independently.

ADC Active Sample Sequencer (ADCACTSS)

Base 0x4003.8000

Offset 0x000

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												ASEN3	ASEN2	ASEN1	ASEN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of 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	Specifies whether Sample Sequencer 3 is enabled. If set, the sample sequence logic for Sequencer 3 is active. Otherwise, the Sequencer is inactive.
2	ASEN2	R/W	0	Specifies whether Sample Sequencer 2 is enabled. If set, the sample sequence logic for Sequencer 2 is active. Otherwise, the Sequencer is inactive.
1	ASEN1	R/W	0	Specifies whether Sample Sequencer 1 is enabled. If set, the sample sequence logic for Sequencer 1 is active. Otherwise, the Sequencer is inactive.
0	ASEN0	R/W	0	Specifies whether Sample Sequencer 0 is enabled. If set, the sample sequence logic for Sequencer 0 is active. Otherwise, the Sequencer is inactive.

Register 2: ADC Raw Interrupt Status (ADCRIS), offset 0x004

This register shows the status of the raw interrupt signal of each Sample Sequencer. These bits may be polled by software to look for interrupt conditions without having to generate controller interrupts.

ADC Raw Interrupt Status (ADCRIS)

Base 0x4003.8000
 Offset 0x004
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												INR3	INR2	INR1	INR0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	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	INR3	RO	0	Set by hardware when a sample with its respective ADCSSCTL3 <small>IE</small> bit has completed conversion. This bit is cleared by writing a 1 to the ADCISC <small>IN3</small> bit.
2	INR2	RO	0	Set by hardware when a sample with its respective ADCSSCTL2 <small>IE</small> bit has completed conversion. This bit is cleared by writing a 1 to the ADCISC <small>IN2</small> bit.
1	INR1	RO	0	Set by hardware when a sample with its respective ADCSSCTL1 <small>IE</small> bit has completed conversion. This bit is cleared by writing a 1 to the ADCISC <small>IN1</small> bit.
0	INR0	RO	0	Set by hardware when a sample with its respective ADCSSCTL0 <small>IE</small> bit has completed conversion. This bit is cleared by writing a 1 to the ADCISC <small>IN0</small> bit.

Register 3: ADC Interrupt Mask (ADCIM), offset 0x008

This register controls whether the Sample Sequencer raw interrupt signals are promoted to controller interrupts. The raw interrupt signal for each Sample Sequencer can be masked independently.

ADC Interrupt Mask (ADCIM)

Base 0x4003.8000
Offset 0x008
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												MASK3	MASK2	MASK1	MASK0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of 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	Specifies whether the raw interrupt signal from Sample Sequencer 3 (ADCRIS register <code>INR3</code> bit) is promoted to a controller interrupt. If set, the raw interrupt signal is promoted to a controller interrupt. Otherwise, it is not.
2	MASK2	R/W	0	Specifies whether the raw interrupt signal from Sample Sequencer 2 (ADCRIS register <code>INR2</code> bit) is promoted to a controller interrupt. If set, the raw interrupt signal is promoted to a controller interrupt. Otherwise, it is not.
1	MASK1	R/W	0	Specifies whether the raw interrupt signal from Sample Sequencer 1 (ADCRIS register <code>INR1</code> bit) is promoted to a controller interrupt. If set, the raw interrupt signal is promoted to a controller interrupt. Otherwise, it is not.
0	MASK0	R/W	0	Specifies whether the raw interrupt signal from Sample Sequencer 0 (ADCRIS register <code>INR0</code> bit) is promoted to a controller interrupt. If set, the raw interrupt signal is promoted to a controller interrupt. Otherwise, it is not.

Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C

This register provides the mechanism for clearing interrupt conditions, and shows the status of controller interrupts generated by the Sample Sequencers. When read, each bit field is the logical AND of the respective **INR** and **MASK** bits. Interrupts are cleared by writing a 1 to the corresponding bit position. If software is polling the **ADCRIS** instead of generating interrupts, the **INR** bits are still cleared via the **ADCISC** register, even if the **IN** bit is not set.

ADC Interrupt Status and Clear (ADCISC)

Base 0x4003.8000
 Offset 0x00C
 Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												IN3	IN2	IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0	Software should not rely on the value of 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	This bit is set by hardware when the MASK3 and INR3 bits are both 1, providing a level-based interrupt to the controller. It is cleared by writing a 1, and also clears the INR3 bit.
2	IN2	R/W1C	0	This bit is set by hardware when the MASK2 and INR2 bits are both 1, providing a level based interrupt to the controller. It is cleared by writing a 1, and also clears the INR2 bit.
1	IN1	R/W1C	0	This bit is set by hardware when the MASK1 and INR1 bits are both 1, providing a level based interrupt to the controller. It is cleared by writing a 1, and also clears the INR1 bit.
0	IN0	R/W1C	0	This bit is set by hardware when the MASK0 and INR0 bits are both 1, providing a level based interrupt to the controller. It is cleared by writing a 1, and also clears the INR0 bit.

Register 5: ADC Overflow Status (ADCOSTAT), offset 0x010

This register indicates overflow conditions in the Sample Sequencer FIFOs. Once the overflow condition has been handled by software, the condition can be cleared by writing a 1 to the corresponding bit position.

ADC Overflow Status (ADCOSTAT)

Base 0x4003.8000

Offset 0x010

Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												OV3	OV2	OV1	OV0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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

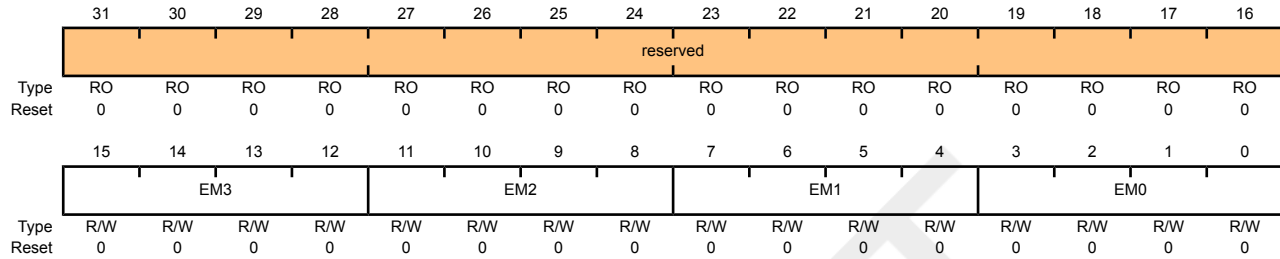
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OV3	R/W1C	0	This bit specifies that the FIFO for Sample Sequencer 3 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped and this bit is set by hardware to indicate the occurrence of dropped data. This bit is cleared by writing a 1.
2	OV2	R/W1C	0	This bit specifies that the FIFO for Sample Sequencer 2 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped and this bit is set by hardware to indicate the occurrence of dropped data. This bit is cleared by writing a 1.
1	OV1	R/W1C	0	This bit specifies that the FIFO for Sample Sequencer 1 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped and this bit is set by hardware to indicate the occurrence of dropped data. This bit is cleared by writing a 1.
0	OV0	R/W1C	0	This bit specifies that the FIFO for Sample Sequencer 0 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped and this bit is set by hardware to indicate the occurrence of dropped data. This bit is cleared by writing a 1.

Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014

The **ADCEMUX** selects the event (trigger) that initiates sampling for each Sample Sequencer. Each Sample Sequencer can be configured with a unique trigger source.

ADC Event Multiplexer Select (ADCEMUX)

Base 0x4003.8000
 Offset 0x014
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description																								
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.																								
15:12	EM3	R/W	0	This field selects the trigger source for Sample Sequencer 3. The valid configurations for this field are: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>EM Binary Value</th> <th>Event</th> </tr> </thead> <tbody> <tr><td>0000</td><td>Controller (default)</td></tr> <tr><td>0001</td><td>Analog Comparator 0</td></tr> <tr><td>0010</td><td>Analog Comparator 1</td></tr> <tr><td>0011</td><td>Analog Comparator 2</td></tr> <tr><td>0100</td><td>External (GPIO PB4)</td></tr> <tr><td>0101</td><td>Timer</td></tr> <tr><td>0110</td><td>Reserved</td></tr> <tr><td>0111</td><td>Reserved</td></tr> <tr><td>1000</td><td>Reserved</td></tr> <tr><td>1001-1110</td><td>reserved</td></tr> <tr><td>1111</td><td>Always (continuously sample)</td></tr> </tbody> </table>	EM Binary Value	Event	0000	Controller (default)	0001	Analog Comparator 0	0010	Analog Comparator 1	0011	Analog Comparator 2	0100	External (GPIO PB4)	0101	Timer	0110	Reserved	0111	Reserved	1000	Reserved	1001-1110	reserved	1111	Always (continuously sample)
EM Binary Value	Event																											
0000	Controller (default)																											
0001	Analog Comparator 0																											
0010	Analog Comparator 1																											
0011	Analog Comparator 2																											
0100	External (GPIO PB4)																											
0101	Timer																											
0110	Reserved																											
0111	Reserved																											
1000	Reserved																											
1001-1110	reserved																											
1111	Always (continuously sample)																											
11:8	EM2	R/W	0	This field selects the trigger source for Sample Sequencer 2. The encodings are the same as those for EM3.																								
7:4	EM1	R/W	0	This field selects the trigger source for Sample Sequencer 1. The encodings are the same as those for EM3.																								
3:0	EM0	R/W	0	This field selects the trigger source for Sample Sequencer 0. The encodings are the same as those for EM3.																								

Register 7: ADC Underflow Status (ADCUSTAT), offset 0x018

This register indicates underflow conditions in the Sample Sequencer FIFOs. The corresponding underflow condition can be cleared by writing a 1 to the relevant bit position.

ADC Underflow Status (ADCUSTAT)

Base 0x4003.8000

Offset 0x018

Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												UV3	UV2	UV1	UV0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of 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	This bit specifies that the FIFO for Sample Sequencer 3 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.
2	UV2	R/W1C	0	This bit specifies that the FIFO for Sample Sequencer 2 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.
1	UV1	R/W1C	0	This bit specifies that the FIFO for Sample Sequencer 1 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.
0	UV0	R/W1C	0	This bit specifies that the FIFO for Sample Sequencer 0 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.

Register 8: ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020

This register sets the priority for each of the Sample Sequencers. Out of reset, Sequencer 0 has the highest priority, and sample sequence 3 has the lowest priority. When reconfiguring sequence priorities, each sequence must have a unique priority or the ADC behavior is inconsistent.

ADC Sample Sequencer Priority (ADCSSPRI)

Base 0x4003.8000
 Offset 0x020
 Type R/W, reset 0x0000.3210

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		SS3		reserved		SS2		reserved		SS1		reserved		SS0	
Type	RO	RO	R/W	R/W	RO	RO	R/W	R/W	RO	RO	R/W	R/W	RO	RO	R/W	R/W
Reset	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0	Software should not rely on the value of 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	The SS3 field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 3. A priority encoding of 0 is highest and 3 is lowest. The priorities assigned to the Sequencers must be uniquely mapped. ADC behavior is not consistent if two or more fields are equal.
11:10	reserved	RO	0	Software should not rely on the value of 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	The SS2 field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 2.
7:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:4	SS1	R/W	0x1	The SS1 field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 1.
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	SS0	R/W	0x0	The SS0 field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 0.

Register 9: ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028

This register provides a mechanism for application software to initiate sampling in the Sample Sequencers. Sample sequences can be initiated individually or in any combination. When multiple sequences are triggered simultaneously, the priority encodings in **ADCSSPRI** dictate execution order.

ADC Processor Sample Sequence Initiate (ADCPSSI)

Base 0x4003.8000

Offset 0x028

Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												SS3	SS2	SS1	SS0
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:4	reserved	WO	-	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	SS3	WO	-	Only a write by software is valid; a read of the register returns no meaningful data. When set by software, sampling is triggered on Sample Sequencer 3, assuming the Sequencer is enabled in the ADCACTSS register.
2	SS2	WO	-	Only a write by software is valid; a read of the register returns no meaningful data. When set by software, sampling is triggered on Sample Sequencer 2, assuming the Sequencer is enabled in the ADCACTSS register.
1	SS1	WO	-	Only a write by software is valid; a read of the register returns no meaningful data. When set by software, sampling is triggered on Sample Sequencer 1, assuming the Sequencer is enabled in the ADCACTSS register.
0	SS0	WO	-	Only a write by software is valid; a read of the register returns no meaningful data. When set by software, sampling is triggered on Sample Sequencer 0, assuming the Sequencer is enabled in the ADCACTSS register.

Register 10: ADC Sample Averaging Control (ADCSAC), offset 0x030

This register controls the amount of hardware averaging applied to conversion results. The final conversion result stored in the FIFO is averaged from 2^{AVG} consecutive ADC samples at the specified ADC speed. If AVG is 0, the sample is passed directly through without any averaging. If AVG=6, then 64 consecutive ADC samples are averaged to generate one result in the sequencer FIFO. An AVG = 7 provides unpredictable results.

ADC Sample Averaging Control (ADCSAC)

Base 0x4003.8000
 Offset 0x030
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													AVG		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0	Software should not rely on the value of 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	0	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.

Register 11: ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 0.

This register is 32-bits wide and contains information for eight possible samples.

ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0)

Base 0x4003.8000
Offset 0x040
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	MUX7			reserved	MUX6			reserved	MUX5			reserved	MUX4		
Type	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	MUX3			reserved	MUX2			reserved	MUX1			reserved	MUX0		
Type	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30:28	MUX7	R/W	0	The MUX7 field is used during the eighth sample of a sequence executed with the Sample Sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. The value set here indicates the corresponding pin, for example, a value of 1 indicates the input is ADC1.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26:24	MUX6	R/W	0	The MUX6 field is used during the seventh sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
23	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22:20	MUX5	R/W	0	The MUX5 field is used during the sixth sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
19	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
18:16	MUX4	R/W	0	The MUX4 field is used during the fifth sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.

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.
14:12	MUX3	R/W	0	The MUX3 field is used during the fourth sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10:8	MUX2	R/W	0	The MUX2 field is used during the third sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:4	MUX1	R/W	0	The MUX1 field is used during the second sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	MUX0	R/W	0	The MUX0 field is used during the first sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.

Register 12: ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044

This register contains the configuration information for each sample for a sequence executed with Sample Sequencer 0. When configuring a sample sequence, the `END` bit must be set at some point, whether it be after the first sample, last sample, or any sample in between.

This register is 32-bits wide and contains information for eight possible samples.

ADC Sample Sequence Control 0 (ADCSSCTL0)

Base 0x4003.8000
Offset 0x044
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	TS7	R/W	0	The <code>TS7</code> bit is used during the eighth sample of the sample sequence and specifies the input source of the sample. If set, the temperature sensor is read. Otherwise, the input pin specified by the <code>ADCSSMUX</code> register is read.
30	IE7	R/W	0	The <code>IE7</code> bit is used during the eighth sample of the sample sequence and specifies whether the raw interrupt signal (<code>INR0</code> bit) is asserted at the end of the sample's conversion. If the <code>MASK0</code> bit in the <code>ADCIM</code> register is set, the interrupt is promoted to a controller-level interrupt. When this bit is set, the raw interrupt is asserted, otherwise it is not. It is legal to have multiple samples within a sequence generate interrupts.
29	END7	R/W	0	The <code>END7</code> bit indicates that this is the last sample of the sequence. It is possible to end the sequence on any sample position. Samples defined after the sample containing a set <code>END</code> are not requested for conversion even though the fields may be non-zero. It is required that software write the <code>END</code> bit somewhere within the sequence. (Sample Sequencer 3, which only has a single sample in the sequence, is hardwired to have the <code>END0</code> bit set.) Setting this bit indicates that this sample is the last in the sequence.
28	D7	R/W	0	The <code>D7</code> bit indicates that the analog input is to be differentially sampled. The corresponding <code>ADCSSMUXx</code> nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". The temperature sensor does not have a differential option. When set, the analog inputs are differentially sampled.
27	TS6	R/W	0	Same definition as <code>TS7</code> but used during the seventh sample.
26	IE6	R/W	0	Same definition as <code>IE7</code> but used during the seventh sample.
25	END6	R/W	0	Same definition as <code>END7</code> but used during the seventh sample.
24	D6	R/W	0	Same definition as <code>D7</code> but used during the seventh sample.
23	TS5	R/W	0	Same definition as <code>TS7</code> but used during the sixth sample.

Bit/Field	Name	Type	Reset	Description
22	IE5	R/W	0	Same definition as IE7 but used during the sixth sample.
21	END5	R/W	0	Same definition as END7 but used during the sixth sample.
20	D5	R/W	0	Same definition as D7 but used during the sixth sample.
19	TS4	R/W	0	Same definition as TS7 but used during the fifth sample.
18	IE4	R/W	0	Same definition as IE7 but used during the fifth sample.
17	END4	R/W	0	Same definition as END7 but used during the fifth sample.
16	D4	R/W	0	Same definition as D7 but used during the fifth sample.
15	TS3	R/W	0	Same definition as TS7 but used during the fourth sample.
14	IE3	R/W	0	Same definition as IE7 but used during the fourth sample.
13	END3	R/W	0	Same definition as END7 but used during the fourth sample.
12	D3	R/W	0	Same definition as D7 but used during the fourth sample.
11	TS2	R/W	0	Same definition as TS7 but used during the third sample.
10	IE2	R/W	0	Same definition as IE7 but used during the third sample.
9	END2	R/W	0	Same definition as END7 but used during the third sample.
8	D2	R/W	0	Same definition as D7 but used during the third sample.
7	TS1	R/W	0	Same definition as TS7 but used during the second sample.
6	IE1	R/W	0	Same definition as IE7 but used during the second sample.
5	END1	R/W	0	Same definition as END7 but used during the second sample.
4	D1	R/W	0	Same definition as D7 but used during the second sample.
3	TS0	R/W	0	Same definition as TS7 but used during the first sample.
2	IE0	R/W	0	Same definition as IE7 but used during the first sample.
1	END0	R/W	0	Same definition as END7 but used during the first sample. Since this sequencer has only one entry, this bit must be set.
0	D0	R/W	0	Same definition as D7 but used during the first sample.

Register 13: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048

Register 14: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068

Register 15: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088

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, and **ADCSSFIFO2** for Sequencer 2). Reads of this register return conversion result data in the order sample 0, sample 1, and so on, until the FIFO is empty. If the FIFO is not properly handled by software, overflow and underflow conditions are registered in the **ADCOSTAT** and **ADCUSTAT** registers.

ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0)

Base 0x4003.8000

Offset 0x048

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0	Software should not rely on the value of 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	0	Conversion result data.

Register 16: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C

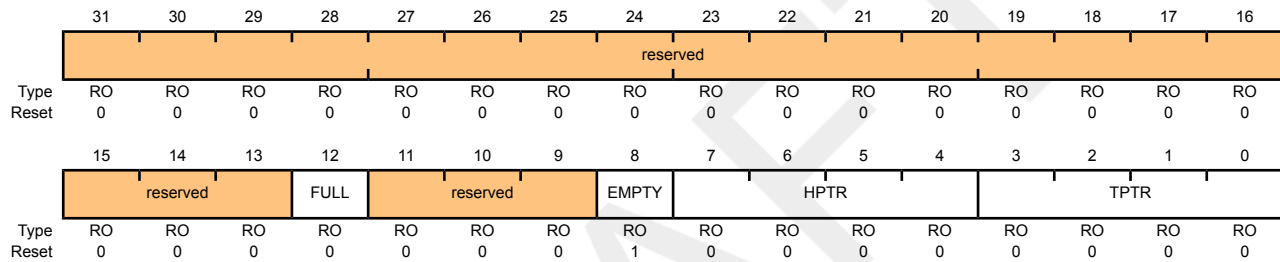
Register 17: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C

Register 18: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C

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 FIFO, **ADCSSFSTAT1** on FIFO1, and **ADCSSFSTAT2** on FIFO2.

ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0)

Base 0x4003.8000
 Offset 0x04C
 Type RO, reset 0x0000.0100



Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0	Software should not rely on the value of 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	When set, indicates that the FIFO is currently full.
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	EMPTY	RO	1	When set, indicates that the FIFO is currently empty.
7:4	HPTR	RO	0	This field contains the current "head" pointer index for the FIFO, that is, the next entry to be written.
3:0	TPTR	RO	0	This field contains the current "tail" pointer index for the FIFO, that is, the next entry to be read.

Register 19: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 1. This register is 16-bits wide and contains information for four possible samples.

ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1)

Base 0x4003.8000
Offset 0x060
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	MUX3			reserved	MUX2			reserved	MUX1			reserved	MUX0		
Type	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14:12	MUX3	R/W	0	The MUX3 field is used during the fourth sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10:8	MUX2	R/W	0	The MUX2 field is used during the third sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:4	MUX1	R/W	0	The MUX1 field is used during the second sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	MUX0	R/W	0	The MUX0 field is used during the first sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.

Register 20: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064

This register contains the configuration information for each sample for a sequence executed with Sample Sequencer 1. When configuring a sample sequence, the `END` bit must be set at some point, whether it be after the first sample, last sample, or any sample in between. This register is 16-bits wide and contains information for four possible samples.

ADC Sample Sequence Control 1 (ADCSSCTL1)

Base 0x4003.8000
 Offset 0x064
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of 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	Same definition as <code>TS7</code> but used during the fourth sample.
14	IE3	R/W	0	Same definition as <code>IE7</code> but used during the fourth sample.
13	END3	R/W	0	Same definition as <code>END7</code> but used during the fourth sample.
12	D3	R/W	0	Same definition as <code>D7</code> but used during the fourth sample.
11	TS2	R/W	0	Same definition as <code>TS7</code> but used during the third sample.
10	IE2	R/W	0	Same definition as <code>IE7</code> but used during the third sample.
9	END2	R/W	0	Same definition as <code>END7</code> but used during the third sample.
8	D2	R/W	0	Same definition as <code>D7</code> but used during the third sample.
7	TS1	R/W	0	Same definition as <code>TS7</code> but used during the second sample.
6	IE1	R/W	0	Same definition as <code>IE7</code> but used during the second sample.
5	END1	R/W	0	Same definition as <code>END7</code> but used during the second sample.
4	D1	R/W	0	Same definition as <code>D7</code> but used during the second sample.
3	TS0	R/W	0	Same definition as <code>TS7</code> but used during the first sample.
2	IE0	R/W	0	Same definition as <code>IE7</code> but used during the first sample.
1	END0	R/W	0	Same definition as <code>END7</code> but used during the first sample. Since this sequencer has only one entry, this bit must be set.
0	D0	R/W	0	Same definition as <code>D7</code> but used during the first sample.

Register 21: 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 2. This register is 16-bits wide and contains information for four possible samples.

ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2)

Base 0x4003.8000
Offset 0x080
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	MUX3			reserved	MUX2			reserved	MUX1			reserved	MUX0		
Type	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14:12	MUX3	R/W	0	The MUX3 field is used during the fourth sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10:8	MUX2	R/W	0	The MUX2 field is used during the third sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:4	MUX1	R/W	0	The MUX1 field is used during the second sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	MUX0	R/W	0	The MUX0 field is used during the first sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.

Register 22: ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084

This register contains the configuration information for each sample for a sequence executed with Sample Sequencer 2. When configuring a sample sequence, the `END` bit must be set at some point, whether it be after the first sample, last sample, or any sample in between. This register is 16-bits wide and contains information for four possible samples.

ADC Sample Sequence Control 2 (ADCSSCTL2)

Base 0x4003.8000
 Offset 0x084
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of 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	Same definition as <code>TS7</code> but used during the fourth sample.
14	IE3	R/W	0	Same definition as <code>IE7</code> but used during the fourth sample.
13	END3	R/W	0	Same definition as <code>END7</code> but used during the fourth sample.
12	D3	R/W	0	Same definition as <code>D7</code> but used during the fourth sample.
11	TS2	R/W	0	Same definition as <code>TS7</code> but used during the third sample.
10	IE2	R/W	0	Same definition as <code>IE7</code> but used during the third sample.
9	END2	R/W	0	Same definition as <code>END7</code> but used during the third sample.
8	D2	R/W	0	Same definition as <code>D7</code> but used during the third sample.
7	TS1	R/W	0	Same definition as <code>TS7</code> but used during the second sample.
6	IE1	R/W	0	Same definition as <code>IE7</code> but used during the second sample.
5	END1	R/W	0	Same definition as <code>END7</code> but used during the second sample.
4	D1	R/W	0	Same definition as <code>D7</code> but used during the second sample.
3	TS0	R/W	0	Same definition as <code>TS7</code> but used during the first sample.
2	IE0	R/W	0	Same definition as <code>IE7</code> but used during the first sample.
1	END0	R/W	0	Same definition as <code>END7</code> but used during the first sample. Since this sequencer has only one entry, this bit must be set.
0	D0	R/W	0	Same definition as <code>D7</code> but used during the first sample.

Register 23: ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 3. This register is 4-bits wide and contains information for one possible sample.

ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3)

Base 0x4003.8000

Offset 0x0A0

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													MUX0		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	MUX0	R/W	0	The MUX0 field is used during the first sample of a sequence executed with the Sample Sequencer and specifies which of the analog inputs is sampled for the analog-to-digital conversion.

Register 24: ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4

This register contains the configuration information for each sample for a sequence executed with Sample Sequencer 3. The `END` bit is always set since there is only one sample in this sequencer. This register is 4-bits wide and contains information for one possible sample.

ADC Sample Sequence Control 3 (ADCSSCTL3)

Base 0x4003.8000
 Offset 0x0A4
 Type R/W, reset 0x0000.0002

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												TS0	IE0	END0	D0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TS0	R/W	0	Same definition as <code>TS7</code> but used during the first sample.
2	IE0	R/W	0	Same definition as <code>IE7</code> but used during the first sample.
1	END0	R/W	0	Same definition as <code>END7</code> but used during the first sample. Since this sequencer has only one entry, this bit must be set.
0	D0	R/W	0	Same definition as <code>D7</code> but used during the first sample.

Register 25: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8

This register contains the conversion results for samples collected with Sample Sequencer 3. Reads of this register return the conversion result data. If the FIFO is not properly handled by software, overflow and underflow conditions are registered in the **ADCOSTAT** and **ADCUSTAT** registers.

Bit fields and definitions are the same as **ADCSSFIFO0** (see page 273) but are for FIFO 3.

DRAFT

Register 26: ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC

This register provides a window into the Sample Sequencer FIFO 3, 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.

This register has the same bit fields and definitions as **ADCSSFSTAT0** (see page 274) but is for FIFO 3.

DRAFT

Register 27: ADC Test Mode Loopback (ADCTMLB), offset 0x100

This register provides loopback operation within the digital logic of the ADC, which can be useful in debugging software without having to provide actual analog stimulus. This test mode is entered by writing a value of 0x0000.0001 to this register. When data is read from the FIFO in loopback mode, the read-only portion of this register is returned.

Read-Only Register

ADC Test Mode Loopback (ADCTMLB)

Base 0x4003.8000
Offset 0x100
Type 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							CNT		CONT	DIFF	TS	MUX			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:6	CNT	RO	0	Continuous sample counter that is initialized to 0 and counts each sample as it processed. This helps provide a unique value for the data received.
5	CONT	RO	0	When set, indicates that this is a continuation sample. For example if two sequencers were to run back-to-back, this indicates that the controller kept continuously sampling at full rate.
4	DIFF	RO	0	When set, indicates that this is a differential sample.
3	TS	RO	0	When set, indicates that this is a temperature sensor sample.
2:0	MUX	RO	0	Indicates which analog input is to be sampled.

Write-Only Register

ADC Test Mode Loopback (ADCTMLB)

Base 0x4003.8000
Offset 0x100
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															LB
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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	LB	WO	0	When set, forces a loopback within the digital block to provide information on input and unique numbering. The 10-bit loopback data is defined as shown in the read for bits 9:0 below.

DRAFT

13 Universal Asynchronous Receivers/Transmitters (UARTs)

UART

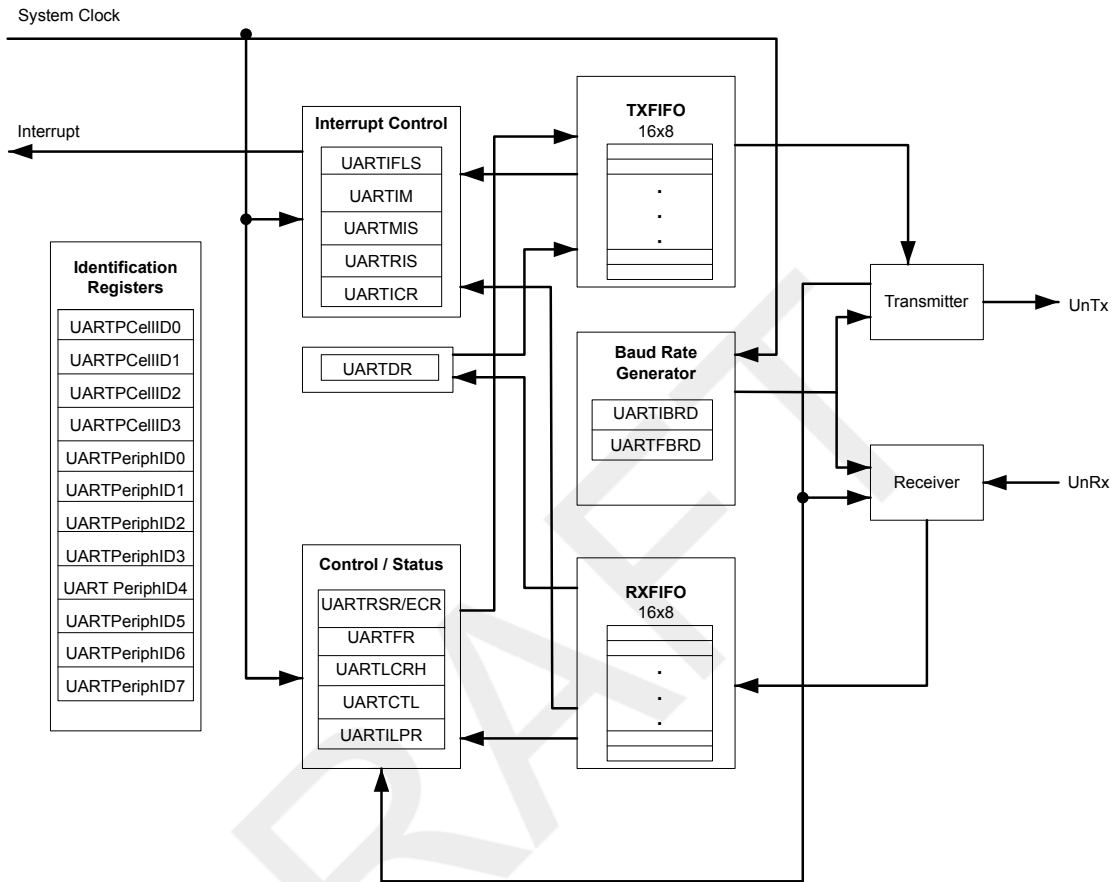
The Stellaris® Universal Asynchronous Receiver/Transmitter (UART) provides fully programmable, 16C550-type serial interface characteristics. The LM3S8938 controller is equipped with three UART modules.

Each UART has the following features:

- Separate transmit and receive FIFOs
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
- Programmable baud-rate generator allowing rates up to 460.8 Kbps
- Standard asynchronous communication bits for start, stop and parity
- False start bit detection
- Line-break generation and detection
- Fully programmable serial interface characteristics:
 - 5, 6, 7, or 8 data bits
 - Even, odd, stick, or no-parity bit generation/detection
 - 1 or 2 stop bit generation
- IrDA serial-IR (SIR) encoder/decoder providing:
 - Programmable use of IrDA Serial InfraRed (SIR) or UART input/output
 - Support of IrDA SIR encoder/decoder functions for data rates up to 115.2 Kbps half-duplex
 - Support of normal 3/16 and low-power (1.41-2.23 μ s) bit durations
 - Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration

13.1 Block Diagram

Figure 13-1. UART Module Block Diagram



13.2 Functional Description

Each Stellaris[®] UART performs the functions of parallel-to-serial and serial-to-parallel conversions. It is similar in functionality to a 16C550 UART, but is not register compatible.

The UART is configured for transmit and/or receive via the **TXE** and **RXE** bits of the **UART Control (UARTCTL)** register (see page 304). Transmit and receive are both enabled out of reset. Before any control registers are programmed, the UART must be disabled by clearing the **UARTEN** bit in **UARTCTL**. If the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

The UART peripheral also includes a serial IR (SIR) encoder/decoder block that can be connected to an infrared transceiver to implement an IrDA SIR physical layer. The SIR function is programmed using the **UARTCTL** register.

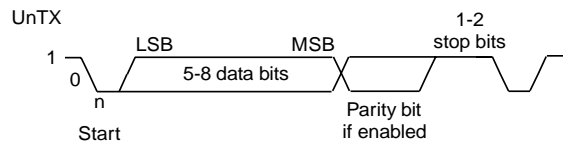
13.2.1 Transmit/Receive Logic

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. The control logic outputs the serial bit stream beginning with a start bit, and followed by the data

bits (LSB first), parity bit, and the stop bits according to the programmed configuration in the control registers. See Figure 13-2 on page 287 for details.

The receive logic performs serial-to-parallel conversion on the received bit stream after a valid start pulse has been detected. Overrun, parity, frame error checking, and line-break detection are also performed, and their status accompanies the data that is written to the receive FIFO.

Figure 13-2. UART Character Frame



13.2.2 Baud-Rate Generation

The baud-rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. The number formed by these two values is used by the baud-rate generator to determine the bit period. Having a fractional baud-rate divider allows the UART to generate all the standard baud rates.

The 16-bit integer is loaded through the **UART Integer Baud-Rate Divisor (UARTIBRD)** register (see page 300) and the 6-bit fractional part is loaded with the **UART Fractional Baud-Rate Divisor (UARTFBRD)** register (see page 301). The baud-rate divisor (BRD) has the following relationship to the system clock (where *BRDI* is the integer part of the BRD and *BRDF* is the fractional part, separated by a decimal place.):

$$BRD = BRDI + BRDF = \text{SysClk} / (16 * \text{Baud Rate})$$

The 6-bit fractional number (that is to be loaded into the *DIVFRAC* bit field in the **UARTFBRD** register) can be calculated by taking the fractional part of the baud-rate divisor, multiplying it by 64, and adding 0.5 to account for rounding errors:

$$\text{UARTFBRD}[\text{DIVFRAC}] = \text{integer}(\text{BRDF} * 64 + 0.5)$$

The UART generates an internal baud-rate reference clock at 16x the baud-rate (referred to as *Baud16*). This reference clock is divided by 16 to generate the transmit clock, and is used for error detection during receive operations.

Along with the **UART Line Control, High Byte (UARTLCRH)** register (see page 302), the **UARTIBRD** and **UARTFBRD** registers form an internal 30-bit register. This internal register is only updated when a write operation to **UARTLCRH** is performed, so any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register for the changes to take effect.

To update the baud-rate registers, there are four possible sequences:

- **UARTIBRD** write, **UARTFBRD** write, and **UARTLCRH** write
- **UARTFBRD** write, **UARTIBRD** write, and **UARTLCRH** write
- **UARTIBRD** write and **UARTLCRH** write
- **UARTFBRD** write and **UARTLCRH** write

13.2.3 Data Transmission

Data received or transmitted is stored in two 16-byte FIFOs, though the receive FIFO has an extra four bits per character for status information. For transmission, data is written into the transmit FIFO. If the UART is enabled, it causes a data frame to start transmitting with the parameters indicated in the **UARTLCRH** register. Data continues to be transmitted until there is no data left in the transmit FIFO. The **BUSY** bit in the **UART Flag (UARTFR)** register (see page 297) is asserted as soon as data is written to the transmit FIFO (that is, if the FIFO is non-empty) and remains asserted while data is being transmitted. The **BUSY** bit is negated only when the transmit FIFO is empty, and the last character has been transmitted from the shift register, including the stop bits. The UART can indicate that it is busy even though the UART may no longer be enabled.

When the receiver is idle (the **UnRx** is continuously 1) and the data input goes Low (a start bit has been received), the receive counter begins running and data is sampled on the eighth cycle of **Baud16** (described in “Transmit/Receive Logic” on page 286).

The start bit is valid if **UnRx** is still low on the eighth cycle of **Baud16**, otherwise a false start bit is detected and it is ignored. Start bit errors can be viewed in the **UART Receive Status (UARTSR)** register (see page 295). If the start bit was valid, successive data bits are sampled on every 16th cycle of **Baud16** (that is, one bit period later) according to the programmed length of the data characters. The parity bit is then checked if parity mode was enabled. Data length and parity are defined in the **UARTLCRH** register.

Lastly, a valid stop bit is confirmed if **UnRx** is High, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO, with any error bits associated with that word.

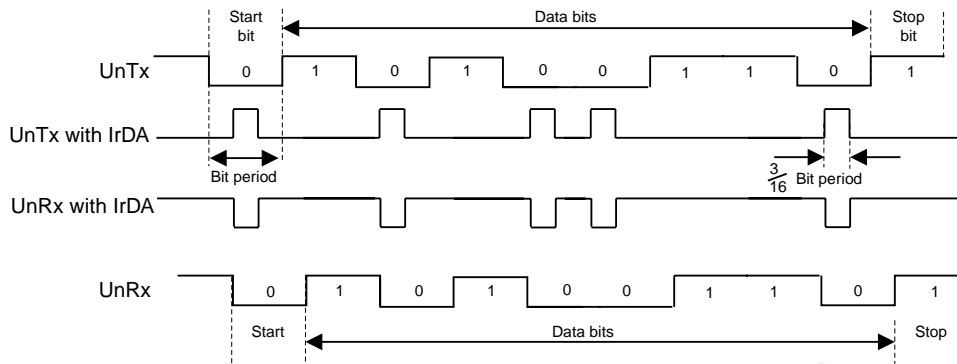
13.2.4 Serial IR (SIR)

The UART peripheral includes an IrDA serial-IR (SIR) encoder/decoder block. The IrDA SIR block provides functionality that converts between an asynchronous UART data stream, and half-duplex serial SIR interface. No analog processing is performed on-chip. The role of the SIR block is to provide a digital encoded output, and decoded input to the UART. The UART signal pins can be connected to an infrared transceiver to implement an IrDA SIR physical layer link. The SIR block has two modes of operation:

- In normal IrDA mode, a zero logic level is transmitted as high pulse of 3/16th duration of the selected baud rate bit period on the output pin, while logic one levels are transmitted as a static LOW signal. These levels control the driver of an infrared transmitter, sending a pulse of light for each zero. On the reception side, the incoming light pulses energize the photo transistor base of the receiver, pulling its output LOW. This drives the UART input pin LOW.
- In low-power IrDA mode, the width of the transmitted infrared pulse is set to three times the period of the internally generated **IrLPBaud16** signal (1.63 μ s, assuming a nominal 1.8432 MHz frequency) by changing the appropriate bit in the **UARTCR** register.

Figure 13-3 on page 289 shows the UART transmit and receive signals, with and without IrDA modulation.

Figure 13-3. IrDA Data Modulation



In both normal and low-power IrDA modes:

- During transmission, the UART data bit is used as the base for encoding
- During reception, the decoded bits are transferred to the UART receive logic

The IrDA SIR physical layer specifies a half-duplex communication link, with a minimum 10 ms delay between transmission and reception. This delay must be generated by software because it is not automatically supported by the UART. The delay is required because the infrared receiver electronics might become biased, or even saturated from the optical power coupled from the adjacent transmitter LED. This delay is known as latency, or receiver setup time.

13.2.5 FIFO Operation

The UART has two 16-entry FIFOs; one for transmit and one for receive. Both FIFOs are accessed via the **UART Data (UARTDR)** register (see page 293). 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 302).

FIFO status can be monitored via the **UART Flag (UARTFR)** register (see page 297) 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 306). 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.

13.2.6 Interrupts

The UART can generate interrupts when the following conditions are observed:

- Overrun Error
- Break Error

- Parity Error
- Framing Error
- Receive Timeout
- Transmit (when condition defined in the `TXIFLSEL` bit in the **UARTIFLS** register is met)
- Receive (when condition defined in the `RXIFLSEL` bit in the **UARTIFLS** register is met)

All of the interrupt events are ORed together before being sent to the interrupt controller, so the UART can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine by reading the **UART Masked Interrupt Status (UARTMIS)** register (see page 310).

The interrupt events that can trigger a controller-level interrupt are defined in the **UART Interrupt Mask (UARTIM)** register (see page 307) by setting the corresponding `IM` bit to 1. If interrupts are not used, the raw interrupt status is always visible via the **UART Raw Interrupt Status (UARTRIS)** register (see page 309).

Interrupts are always cleared (for both the **UARTMIS** and **UARTRIS** registers) by setting the corresponding bit in the **UART Interrupt Clear (UARTICR)** register (see page 311).

13.2.7 Loopback Operation

The UART can be placed into an internal loopback mode for diagnostic or debug work. This is accomplished by setting the `LBE` bit in the **UARTCTL** register (see page 304). In loopback mode, data transmitted on `UnTx` is received on the `UnRx` input.

13.2.8 IrDA SIR block

The IrDA SIR block contains an IrDA serial IR (SIR) protocol encoder/decoder. When enabled, the SIR block uses the `UnTx` and `UnRx` pins for the SIR protocol, which should be connected to an IR transceiver.

The SIR block can receive and transmit, but it is only half-duplex so it cannot do both at the same time. Transmission must be stopped before data can be received. The IrDA SIR physical layer specifies a minimum 10-ms delay between transmission and reception.

13.3 Initialization and Configuration

To use the UARTs, the peripheral clock must be enabled by setting the `UART0`, `UART1`, or `UART2` bits in the **RCGC1** register.

This section discusses the steps that are required for using a UART module. For this example, the system clock is assumed to be 20 MHz and the desired UART configuration is:

- 115200 baud rate
- Data length of 8 bits
- One stop bit
- No parity
- FIFOs disabled

- No interrupts

The first thing to consider when programming the UART is the baud-rate divisor (BRD), since the **UARTIBRD** and **UARTFBRD** registers must be written before the **UARTLCRH** register. Using the equation described in “Baud-Rate Generation” on page 287, 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 300) should be set to 10. The value to be loaded into the **UARTFBRD** register (see page 301) 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. Enable the UART by setting the **UARTEN** bit in the **UARTCTL** register.

13.4 Register Map

Table 13-1 on page 291 lists the UART registers. The offset listed is a hexadecimal increment to the register’s address, relative to that UART’s base address:

- UART0: 0x4000.C000
- UART1: 0x4000.D000
- UART2: 0x4000.E000

Note: The UART must be disabled (see the **UARTEN** bit in the **UARTCTL** register on page 304) before any of the control registers are reprogrammed. When the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

Table 13-1. UART Register Map

Offset	Name	Type	Reset	Description	See page
0x000	UARTDR	RO	0x0000.0000	UART Data	293
0x004	UARTSR/UARTCR	R/W	0x0000.0000	UART Receive Status/Error Clear	295
0x018	UARTFR	RO	0x0000.0090	UART Flag	297
0x020	UARTILPR	R/W	0x0000.0000	UART IrDA Low-Power Register	299
0x024	UARTIBRD	R/W	0x0000.0000	UART Integer Baud-Rate Divisor	300
0x028	UARTFBRD	R/W	0x0000.0000	UART Fractional Baud-Rate Divisor	301

Offset	Name	Type	Reset	Description	See page
0x02C	UARTLCRH	R/W	0x0000.0000	UART Line Control	302
0x030	UARTCTL	R/W	0x0000.0300	UART Control	304
0x034	UARTIFLS	R/W	0x0000.0012	UART Interrupt FIFO Level Select	306
0x038	UARTIM	R/W	0x0000.0000	UART Interrupt Mask	307
0x03C	UARTRIS	RO	0x0000.000F	UART Raw Interrupt Status	309
0x040	UARTMIS	RO	0x0000.0000	UART Masked Interrupt Status	310
0x044	UARTICR	W1C	0x0000.0000	UART Interrupt Clear	311
0xFD0	UARTPeriphID4	RO	0x0000.0000	UART Peripheral Identification 4	313
0xFD4	UARTPeriphID5	RO	0x0000.0000	UART Peripheral Identification 5	314
0xFD8	UARTPeriphID6	RO	0x0000.0000	UART Peripheral Identification 6	315
0xFDC	UARTPeriphID7	RO	0x0000.0000	UART Peripheral Identification 7	316
0xFE0	UARTPeriphID0	RO	0x0000.0011	UART Peripheral Identification 0	317
0xFE4	UARTPeriphID1	RO	0x0000.0000	UART Peripheral Identification 1	318
0xFE8	UARTPeriphID2	RO	0x0000.0018	UART Peripheral Identification 2	319
0xFEC	UARTPeriphID3	RO	0x0000.0001	UART Peripheral Identification 3	320
0xFF0	UARTPCellID0	RO	0x0000.000D	UART PrimeCell Identification 0	321
0xFF4	UARTPCellID1	RO	0x0000.00F0	UART PrimeCell Identification 1	322
0xFF8	UARTPCellID2	RO	0x0000.0005	UART PrimeCell Identification 2	323
0xFFC	UARTPCellID3	RO	0x0000.00B1	UART PrimeCell Identification 3	324

13.5 Register Descriptions

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

Register 1: UART Data (UARTDR), offset 0x000

This register is the data register (the interface to the FIFOs).

When FIFOs are enabled, data written to this location is pushed onto the transmit FIFO. If FIFOs are disabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). A write to this register initiates a transmission from the UART.

For received data, if the FIFO is enabled, the data byte and the 4-bit status (break, frame, parity and overrun) is pushed onto the 12-bit wide receive FIFO. If FIFOs are disabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO). The received data can be retrieved by reading this register.

UART Data (UARTDR)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0x000

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	OE	RO	0	UART Overrun Error 1=New data was received when the FIFO was full, resulting in data loss. 0=There has been no data loss due to a FIFO overrun.
10	BE	RO	0	UART Break Error This bit is set to 1 when a break condition is detected, indicating that the receive data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits). In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the received data input goes to a 1 (marking state) and the next valid start bit is received.
9	PE	RO	0	UART Parity Error This bit is set to 1 when the parity of the received data character does not match the parity defined by bits 2 and 7 of the UARTLCRH register. In FIFO mode, this error is associated with the character at the top of the FIFO.

Bit/Field	Name	Type	Reset	Description
8	FE	RO	0	UART Framing Error This bit is set to 1 when the received character does not have a valid stop bit (a valid stop bit is 1).
7:0	DATA	R/W	0	When written, the data that is to be transmitted via the UART. When read, the data that was received by the UART.

DRAFT

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.

A write of any value to the **UARTECR** register clears the framing, parity, break, and overrun errors. All the bits are cleared to 0 on reset.

Read-Only Receive Status (UARTRSR) Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x004
 Type 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
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

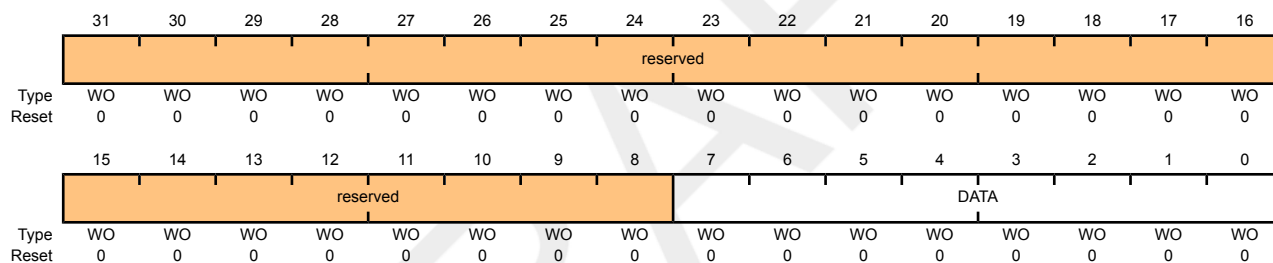
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. The UARTRSR register cannot be written.
3	OE	RO	0	UART Overrun Error When this bit is set to 1, data is received and the FIFO is already full. This bit is cleared to 0 by a write to UARTECR . The FIFO contents remain valid since no further data is written when the FIFO is full, only the contents of the shift register are overwritten. The CPU must now read the data in order to empty the FIFO.
2	BE	RO	0	UART Break Error This bit is set to 1 when a break condition is detected, indicating that the received data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits). This bit is cleared to 0 by a write to 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.

Bit/Field	Name	Type	Reset	Description
1	PE	RO	0	<p>UART Parity Error</p> <p>This bit is set to 1 when the parity of the received data character does not match the parity defined by bits 2 and 7 of the UARTLCRH register.</p> <p>This bit is cleared to 0 by a write to UARTECR.</p>
0	FE	RO	0	<p>UART Framing Error</p> <p>This bit is set to 1 when the received character does not have a valid stop bit (a valid stop bit is 1).</p> <p>This bit is cleared to 0 by a write to UARTECR.</p> <p>In FIFO mode, this error is associated with the character at the top of the FIFO.</p>

Write-Only Error Clear (UARTECR) Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x004
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	WO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	WO	0	A write to this register of any data clears the framing, parity, break and overrun flags.

Register 3: UART Flag (UARTFR), offset 0x018

The **UARTFR** register is the flag register. After reset, the **TXFF**, **RXFF**, and **BUSY** bits are 0, and **TXFE** and **RXFE** bits are 1.

UART Flag (UARTFR)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

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								TXFE	RXFF	TXFF	RXFE	BUSY	reserved		
Type	RO	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
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TXFE	RO	1	<p>UART Transmit FIFO Empty</p> <p>The meaning of this bit depends on the state of the FEN bit in the UARTLCRH register.</p> <p>If the FIFO is disabled (FEN is 0), this bit is set when the transmit holding register is empty.</p> <p>If the FIFO is enabled (FEN is 1), this bit is set when the transmit FIFO is empty.</p>
6	RXFF	RO	0	<p>UART Receive FIFO Full</p> <p>The meaning of this bit depends on the state of the FEN bit in the UARTLCRH register.</p> <p>If the FIFO is disabled, this bit is set when the receive holding register is full.</p> <p>If the FIFO is enabled, this bit is set when the receive FIFO is full.</p>
5	TXFF	RO	0	<p>UART Transmit FIFO Full</p> <p>The meaning of this bit depends on the state of the FEN bit in the UARTLCRH register.</p> <p>If the FIFO is disabled, this bit is set when the transmit holding register is full.</p> <p>If the FIFO is enabled, this bit is set when the transmit FIFO is full.</p>

Bit/Field	Name	Type	Reset	Description
4	RXFE	RO	1	<p>UART Receive FIFO Empty</p> <p>The meaning of this bit depends on the state of the <code>FEN</code> bit in the <code>UARTLCRH</code> register.</p> <p>If the FIFO is disabled, this bit is set when the receive holding register is empty.</p> <p>If the FIFO is enabled, this bit is set when the receive FIFO is empty.</p>
3	BUSY	RO	0	<p>UART Busy</p> <p>When this bit is 1, the UART is busy transmitting data. This bit remains set until the complete byte, including all stop bits, has been sent from the shift register.</p> <p>This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether UART is enabled).</p>
2:0	reserved	RO	0	<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>

DRAFT

Register 4: UART IrDA Low-Power Register (UARTILPR), offset 0x020

The **UARTILPR** register is an 8-bit read/write register that stores the low-power counter divisor value used to generate the $I_{rLPBaud16}$ signal by dividing down the system clock (SysClk). All the bits are cleared to 0 when reset.

The $I_{rLPBaud16}$ internal signal is generated by dividing down the **UARTCLK** signal according to the low-power divisor value written to **UARTILPR**. The low-power divisor value is calculated as follows:

$$ILPDVSR = \text{SysClk} / F_{I_{rLPBaud16}}$$

where $F_{I_{rLPBaud16}}$ is nominally 1.8432 MHz.

$I_{rLPBaud16}$ is an internal signal used for SIR pulse generation when low-power mode is used. You must choose the divisor so that $1.42 \text{ MHz} < F_{I_{rLPBaud16}} < 2.12 \text{ MHz}$, which results in a low-power pulse duration of 1.41–2.11 μs (three times the period of $I_{rLPBaud16}$). The minimum frequency of $I_{rLPBaud16}$ ensures that pulses less than one period of $I_{rLPBaud16}$ are rejected, but that pulses greater than 1.4 μs are accepted as valid pulses.

Note: Zero is an illegal value. Programming a zero value results in no $I_{rLPBaud16}$ pulses being generated.

UART IrDA Low-Power Register (UARTILPR)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x020
 Type R/W, reset 0x0000.0000

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

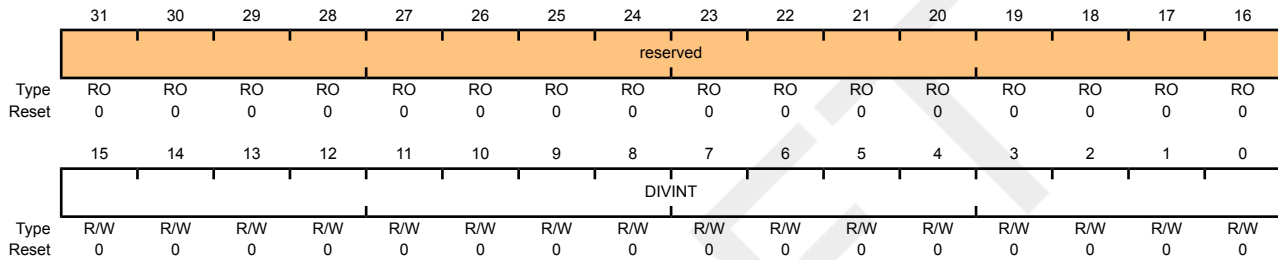
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ILPDVSR	R/W	0x0000	IrDA Low-Power Divisor This is an 8-bit low-power divisor value.

Register 5: UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024

The **UARTIBRD** register is the integer part of the baud-rate divisor value. All the bits are cleared on reset. The minimum possible divide ratio is 1 (when **UARTIBRD**=0), in which case the **UARTFBRD** register is ignored. When changing the **UARTIBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 287 for configuration details.

UART Integer Baud-Rate Divisor (UARTIBRD)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x024
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DIVINT	R/W	0x0000	Integer Baud-Rate Divisor

Register 6: UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028

The **UARTFBRD** register is the fractional part of the baud-rate divisor value. All the bits are cleared on reset. When changing the **UARTFBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 287 for configuration details.

UART Fractional Baud-Rate Divisor (UARTFBRD)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x028
 Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of 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	0x00	Fractional Baud-Rate Divisor

Register 7: UART Line Control (UARTLCRH), offset 0x02C

The **UARTLCRH** register is the line control register. Serial parameters such as data length, parity and stop bit selection are implemented in this register.

When updating the baud-rate divisor (**UARTIBRD** and/or **UARTIFRD**), the **UARTLCRH** register must also be written. The write strobe for the baud-rate divisor registers is tied to the **UARTLCRH** register.

UART Line Control (UARTLCRH)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x02C
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								SPS	WLEN		FEN	STP2	EPS	PEN	BRK
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	SPS	R/W	0	<p>UART Stick Parity Select</p> <p>When bits 1, 2 and 7 of 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.</p> <p>When this bit is cleared, stick parity is disabled.</p>
6:5	WLEN	R/W	0	<p>UART Word Length</p> <p>The bits indicate the number of data bits transmitted or received in a frame as follows:</p> <p>0x3: 8 bits</p> <p>0x2: 7 bits</p> <p>0x1: 6 bits</p> <p>0x0: 5 bits (default)</p>
4	FEN	R/W	0	<p>UART Enable FIFOs</p> <p>If this bit is set to 1, transmit and receive FIFO buffers are enabled (FIFO mode).</p> <p>When cleared to 0, FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.</p>
3	STP2	R/W	0	<p>UART Two Stop Bits Select</p> <p>If this bit is set to 1, two stop bits are transmitted at the end of a frame. The receive logic does not check for two stop bits being received.</p>

Bit/Field	Name	Type	Reset	Description
2	EPS	R/W	0	<p>UART Even Parity Select</p> <p>If this bit is set to 1, even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits.</p> <p>When cleared to 0, then odd parity is performed, which checks for an odd number of 1s.</p> <p>This bit has no effect when parity is disabled by the PEN bit.</p>
1	PEN	R/W	0	<p>UART Parity Enable</p> <p>If this bit is set to 1, parity checking and generation is enabled; otherwise, parity is disabled and no parity bit is added to the data frame.</p>
0	BRK	R/W	0	<p>UART Send Break</p> <p>If this bit is set to 1, a Low level is continually output on the UNTX output, after completing transmission of the current character. For the proper execution of the break command, the software must set this bit for at least two frames (character periods). For normal use, this bit must be cleared to 0.</p>

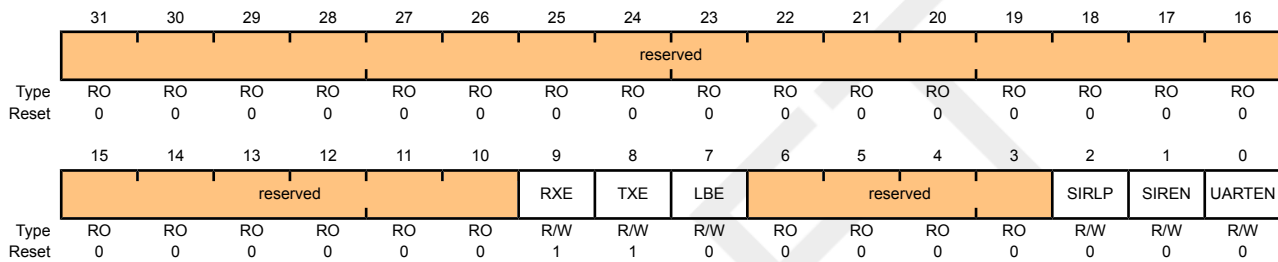
Register 8: UART Control (UARTCTL), offset 0x030

The **UARTCTL** register is the control register. All the bits are cleared on reset except for the Transmit Enable (TXE) and Receive Enable (RXE) bits, which are set to 1.

To enable the UART module, the **UARTEN** bit must be set to 1. If software requires a configuration change in the module, the **UARTEN** bit must be cleared before the configuration changes are written. If the UART is disabled during a transmit or receive operation, the current transaction is completed prior to the UART stopping.

UART Control (UARTCTL)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x030
 Type R/W, reset 0x0000.0300



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	RXE	R/W	1	<p>UART Receive Enable</p> <p>If this bit is set to 1, the receive section of the UART is enabled. When the UART is disabled in the middle of a receive, it completes the current character before stopping.</p> <p>Note: To enable reception, the UARTEN bit must also be set.</p>
8	TXE	R/W	1	<p>UART Transmit Enable</p> <p>If this bit is set to 1, the transmit section of the UART is enabled. When the UART is disabled in the middle of a transmission, it completes the current character before stopping.</p> <p>Note: To enable transmission, the UARTEN bit must also be set.</p>
7	LBE	R/W	0	<p>UART Loop Back Enable</p> <p>If this bit is set to 1, the U_nTX path is fed through the U_nRX path.</p>
6:3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
2	SIRLP	R/W	0	UART SIR Low Power Mode This bit selects the IrDA encoding mode. If this bit is cleared to 0, low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period. If this bit is set to 1, low-level bits are transmitted with a pulse width which is 3 times the period of the $I_{rLPBaud16}$ input signal, regardless of the selected bit rate. Setting this bit uses less power, but might reduce transmission distances. See page 299 for more information.
1	SIREN	R/W	0	UART SIR Enable If this bit is set to 1, the IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol.
0	UARTEN	R/W	0	UART Enable If this bit is set to 1, the UART is enabled. When the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.

Register 9: UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034

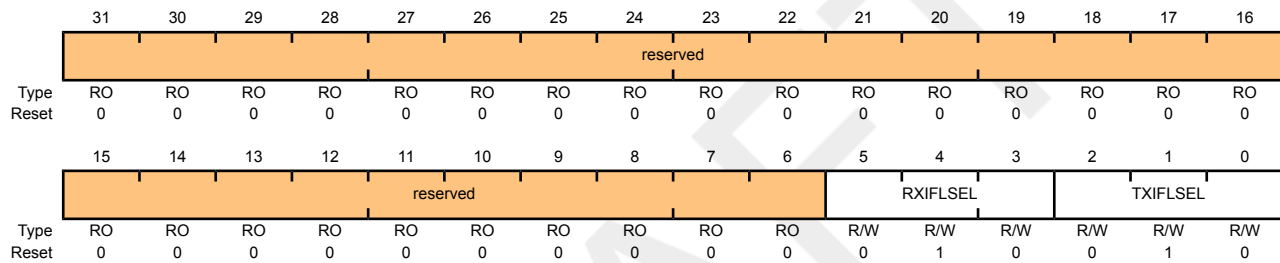
The **UARTIFLS** register is the interrupt FIFO level select register. You can use this register to define the FIFO level at which the **TXRIS** and **RXRIS** bits in the **UARTRIS** register are triggered.

The interrupts are generated based on a transition through a level rather than being based on the level. That is, the interrupts are generated when the fill level progresses through the trigger level. For example, if the receive trigger level is set to the half-way mark, the interrupt is triggered as the module is receiving the 9th character.

Out of reset, the **TXIFLSEL** and **RXIFLSEL** bits are configured so that the FIFOs trigger an interrupt at the half-way mark.

UART Interrupt FIFO Level Select (UARTIFLS)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x034
 Type R/W, reset 0x0000.0012



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of 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: 000: RX FIFO \geq 1/8 full 001: RX FIFO \geq 1/4 full 010: RX FIFO \geq 1/2 full (default) 011: RX FIFO \geq 3/4 full 100: RX FIFO \geq 7/8 full 101-111: Reserved
2:0	TXIFLSEL	R/W	0x2	UART Transmit Interrupt FIFO Level Select The trigger points for the transmit interrupt are as follows: 000: TX FIFO \leq 1/8 full 001: TX FIFO \leq 1/4 full 010: TX FIFO \leq 1/2 full (default) 011: TX FIFO \leq 3/4 full 100: TX FIFO \leq 7/8 full 101-111: Reserved

Register 10: UART Interrupt Mask (UARTIM), offset 0x038

The **UARTIM** register is the interrupt mask set/clear register.

On a read, this register gives the current value of the mask on the relevant interrupt. Writing a 1 to a bit allows the corresponding raw interrupt signal to be routed to the interrupt controller. Writing a 0 prevents the raw interrupt signal from being sent to the interrupt controller.

UART Interrupt Mask (UARTIM)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x038
 Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEIM	R/W	0	UART Overrun Error Interrupt Mask On a read, the current mask for the OEIM interrupt is returned. Setting this bit to 1 promotes the OEIM interrupt to the interrupt controller.
9	BEIM	R/W	0	UART Break Error Interrupt Mask On a read, the current mask for the BEIM interrupt is returned. Setting this bit to 1 promotes the BEIM interrupt to the interrupt controller.
8	PEIM	R/W	0	UART Parity Error Interrupt Mask On a read, the current mask for the PEIM interrupt is returned. Setting this bit to 1 promotes the PEIM interrupt to the interrupt controller.
7	FEIM	R/W	0	UART Framing Error Interrupt Mask On a read, the current mask for the FEIM interrupt is returned. Setting this bit to 1 promotes the FEIM interrupt to the interrupt controller.
6	RTIM	R/W	0	UART Receive Time-Out Interrupt Mask On a read, the current mask for the RTIM interrupt is returned. Setting this bit to 1 promotes the RTIM interrupt to the interrupt controller.
5	TXIM	R/W	0	UART Transmit Interrupt Mask On a read, the current mask for the TXIM interrupt is returned. Setting this bit to 1 promotes the TXIM interrupt to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
4	RXIM	R/W	0	UART Receive Interrupt Mask On a read, the current mask for the <code>RXIM</code> interrupt is returned. Setting this bit to 1 promotes the <code>RXIM</code> interrupt to the interrupt controller.
3:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

DRAFT

Register 11: UART Raw Interrupt Status (UARTRIS), offset 0x03C

The **UARTRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt. A write has no effect.

UART Raw Interrupt Status (UARTRIS)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x03C
 Type RO, reset 0x0000.000F

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

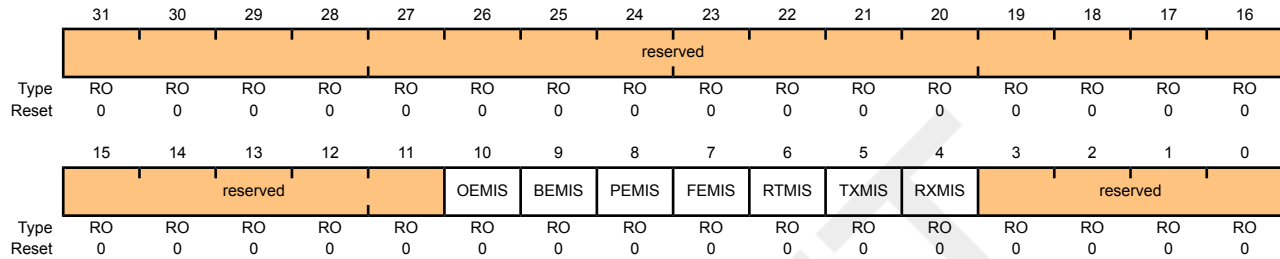
Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OERIS	RO	0	UART Overrun Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
9	BERIS	RO	0	UART Break Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
8	PERIS	RO	0	UART Parity Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
7	FERIS	RO	0	UART Framing Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
6	RTRIS	RO	0	UART Receive Time-Out Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
5	TXRIS	RO	0	UART Transmit Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
4	RXRIS	RO	0	UART Receive Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
3:0	reserved	RO	0xF	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 12: UART Masked Interrupt Status (UARTMIS), offset 0x040

The **UARTMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

UART Masked Interrupt Status (UARTMIS)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x040
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEMIS	RO	0	UART Overrun Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
9	BEMIS	RO	0	UART Break Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
8	PEMIS	RO	0	UART Parity Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
7	FEMIS	RO	0	UART Framing Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
6	RTMIS	RO	0	UART Receive Time-Out Masked Interrupt Status Gives the masked interrupt state of this interrupt.
5	TXMIS	RO	0	UART Transmit Masked Interrupt Status Gives the masked interrupt state of this interrupt.
4	RXMIS	RO	0	UART Receive Masked Interrupt Status Gives the masked interrupt state of this interrupt.
3:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 13: UART Interrupt Clear (UARTICR), offset 0x044

The **UARTICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt (both raw interrupt and masked interrupt, if enabled) is cleared. A write of 0 has no effect.

UART Interrupt Clear (UARTICR)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0x044
 Type W1C, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0	Software should not rely on the value of 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 0: No effect on the interrupt. 1: Clears interrupt.
9	BEIC	W1C	0	Break Error Interrupt Clear 0: No effect on the interrupt. 1: Clears interrupt.
8	PEIC	W1C	0	Parity Error Interrupt Clear 0: No effect on the interrupt. 1: Clears interrupt.
7	FEIC	W1C	0	Framing Error Interrupt Clear 0: No effect on the interrupt. 1: Clears interrupt.
6	RTIC	W1C	0	Receive Time-Out Interrupt Clear 0: No effect on the interrupt. 1: Clears interrupt.
5	TXIC	W1C	0	Transmit Interrupt Clear 0: No effect on the interrupt. 1: Clears interrupt.

Bit/Field	Name	Type	Reset	Description
4	RXIC	W1C	0	Receive Interrupt Clear 0: No effect on the interrupt. 1: Clears interrupt.
3:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

DRAFT

Register 14: UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 4 (UARTPeriphID4)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFD0
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

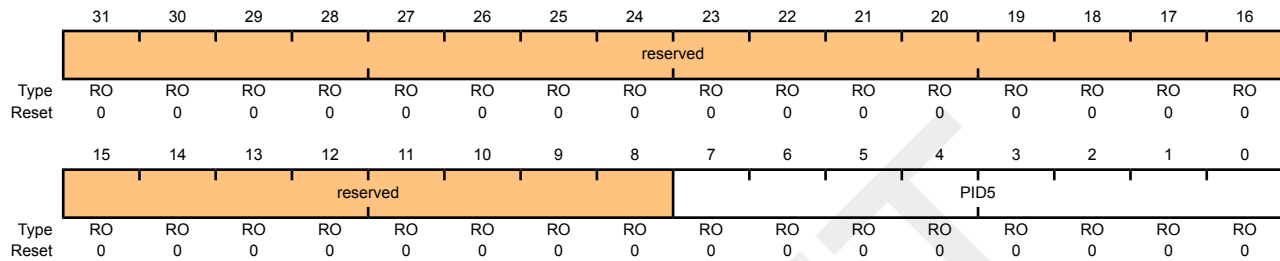
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	UART Peripheral ID Register[7:0]

Register 15: UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 5 (UARTPeriphID5)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFD4
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	UART Peripheral ID Register[15:8]

Register 16: UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 6 (UARTPeriphID6)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFD8
 Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID6							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

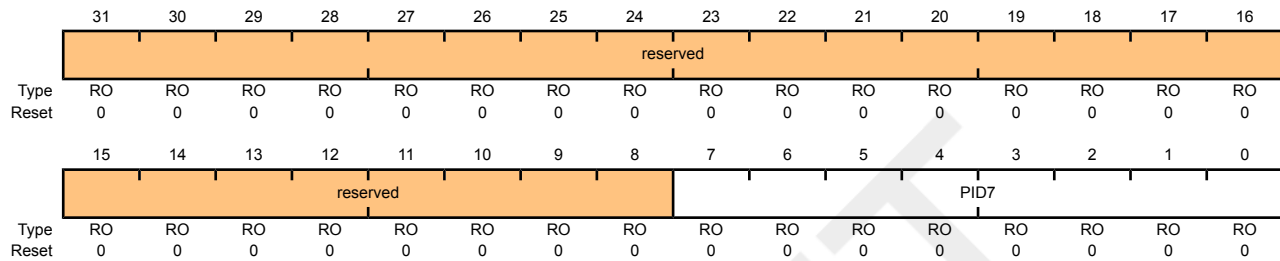
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	UART Peripheral ID Register[23:16]

Register 17: UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 7 (UARTPeriphID7)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFDC
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	UART Peripheral ID Register[31:24]

Register 18: UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 0 (UARTPeriphID0)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFE0
 Type RO, reset 0x0000.0011

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

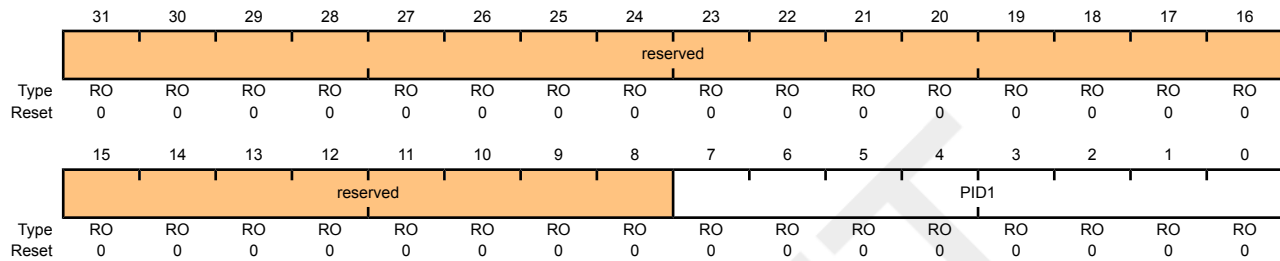
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x11	UART Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 19: UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 1 (UARTPeriphID1)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFE4
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	UART Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

Register 20: UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 2 (UARTPeriphID2)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFE8
 Type RO, reset 0x0000.0018

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

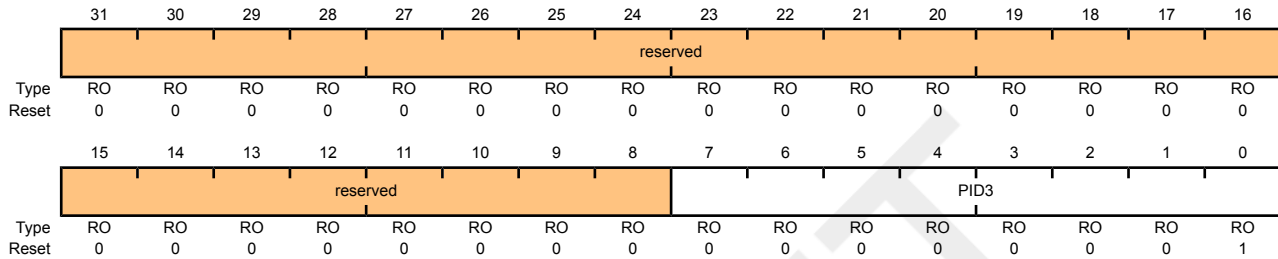
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	UART Peripheral ID Register[23:16] Can be used by software to identify the presence of this peripheral.

Register 21: UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 3 (UARTPeriphID3)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFEC
 Type RO, reset 0x0000.0001



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

Register 22: UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 0 (UARTPCellID0)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFF0
 Type RO, reset 0x0000.000D

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

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

Register 23: UART PrimeCell Identification 1 (UARTPCelIID1), offset 0xFF4

The **UARTPCelIIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 1 (UARTPCelIID1)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFF4
 Type RO, reset 0x0000.00F0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

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

Register 24: UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 2 (UARTPCellID2)

UART0 base: 0x4000.C000

UART1 base: 0x4000.D000

UART2 base: 0x4000.E000

Offset 0xFF8

Type RO, reset 0x0000.0005

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

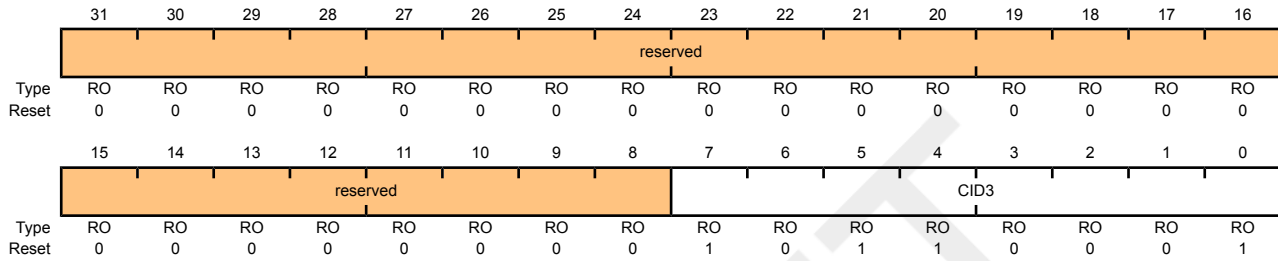
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	UART PrimeCell ID Register[23:16] Provides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

Register 25: UART PrimeCell Identification 3 (UARTPCelIID3), offset 0xFFC

The **UARTPCelIIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 3 (UARTPCelIID3)

UART0 base: 0x4000.C000
 UART1 base: 0x4000.D000
 UART2 base: 0x4000.E000
 Offset 0xFFC
 Type RO, reset 0x0000.00B1



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	UART PrimeCell ID Register[31:24] Provides software a standard cross-peripheral identification system.

14 Synchronous Serial Interface (SSI)

SSI

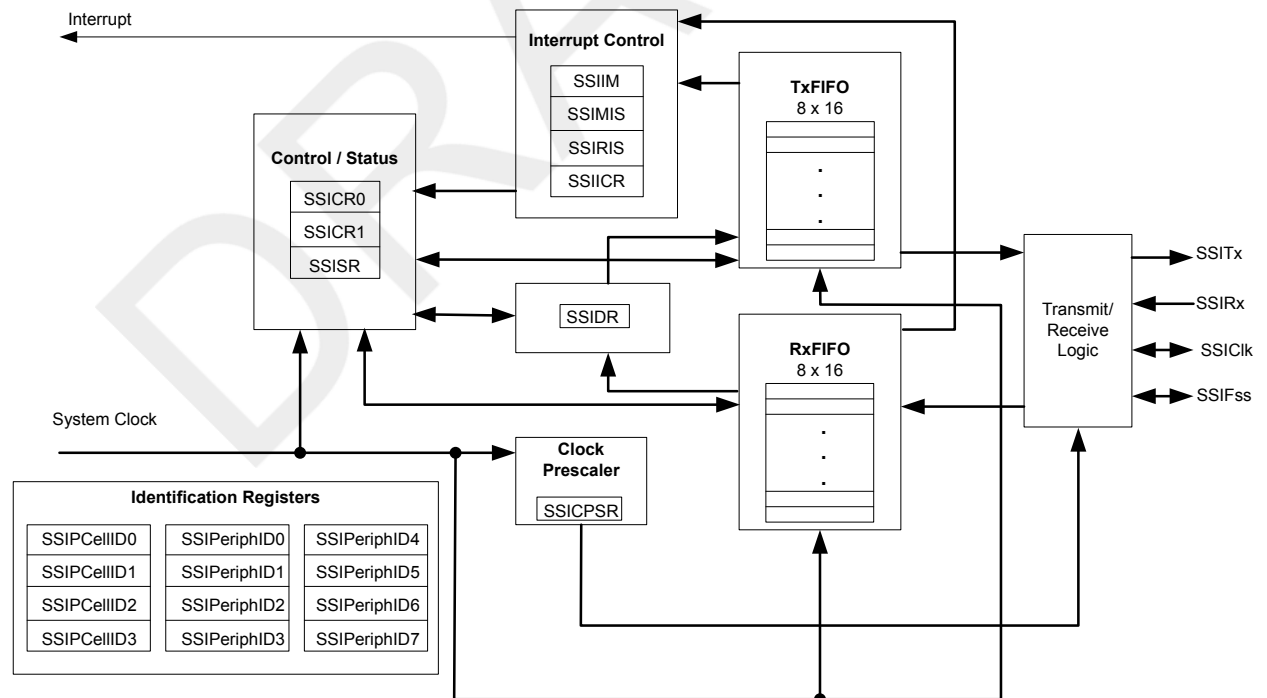
The Stellaris[®] Synchronous Serial Interface (SSI) is a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces.

The Stellaris[®] SSI module has the following features:

- Master or slave operation
- Programmable clock bit rate and prescale
- Separate transmit and receive FIFOs, 16 bits wide, 8 locations deep
- Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces
- Programmable data frame size from 4 to 16 bits
- Internal loopback test mode for diagnostic/debug testing

14.1 Block Diagram

Figure 14-1. SSI Module Block Diagram



14.2 Functional Description

The SSI performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information. The transmit and receive paths are buffered with internal FIFO memories allowing up to eight 16-bit values to be stored independently in both transmit and receive modes.

14.2.1 Bit Rate Generation

The SSI includes a programmable bit rate clock divider and prescaler to generate the serial output clock. Bit rates are supported to 2 MHz and higher, although maximum bit rate is determined by peripheral devices.

The serial bit rate is derived by dividing down the 50-MHz input clock. The clock is first divided by an even prescale value `CPSDVSR` from 2 to 254, which is programmed in the **SSI Clock Prescale (SSICPSR)** register (see page 342). The clock is further divided by a value from 1 to 256, which is $1 + SCR$, where `SCR` is the value programmed in the **SSI Control0 (SSICR0)** register (see page 337).

The frequency of the output clock `SSIClk` is defined by:

$$f_{SSIClk} = f_{SysClk} / (CPSDVSR * (1 + SCR))$$

Note that although the `SSIClk` transmit clock can theoretically be 25 MHz, the module may not be able to operate at that speed. For master mode, the system clock must be at least two times faster than the `SSIClk`. For slave mode, the system clock must be at least 12 times faster than the `SSIClk`.

See “Electrical Characteristics” on page 511 to view SSI timing parameters.

14.2.2 FIFO Operation

14.2.2.1 Transmit FIFO

The common transmit FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. The CPU writes data to the FIFO by writing the **SSI Data (SSIDR)** register (see page 340), and data is stored in the FIFO until it is read out by the transmission logic.

When configured as a master or a slave, parallel data is written into the transmit FIFO prior to serial conversion and transmission to the attached slave or master, respectively, through the `SSITx` pin.

14.2.2.2 Receive FIFO

The common receive FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. Received data from the serial interface is stored in the buffer until read out by the CPU, which accesses the read FIFO by reading the **SSIDR** register.

When configured as a master or slave, serial data received through the `SSIRx` pin is registered prior to parallel loading into the attached slave or master receive FIFO, respectively.

14.2.3 Interrupts

The SSI can generate interrupts when the following conditions are observed:

- Transmit FIFO service
- Receive FIFO service
- Receive FIFO time-out

- Receive FIFO overrun

All of the interrupt events are ORed together before being sent to the interrupt controller, so the SSI can only generate a single interrupt request to the controller at any given time. You can mask each of the four individual maskable interrupts by setting the appropriate bits in the **SSI Interrupt Mask (SSIIM)** register (see page 343). Setting the appropriate mask bit to 1 enables the interrupt.

Provision of the individual outputs, as well as a combined interrupt output, allows use of either a global interrupt service routine, or modular device drivers to handle interrupts. The transmit and receive dynamic dataflow interrupts have been separated from the status interrupts so that data can be read or written in response to the FIFO trigger levels. The status of the individual interrupt sources can be read from the **SSI Raw Interrupt Status (SSIRIS)** and **SSI Masked Interrupt Status (SSIMIS)** registers (see page 344 and page 345, respectively).

14.2.4 Frame Formats

Each data frame is between 4 and 16 bits long, depending on the size of data programmed, and is transmitted starting with the MSB. There are three basic frame types that can be selected:

- Texas Instruments synchronous serial
- Freescale SPI
- MICROWIRE

For all three formats, the serial clock (*SSIClk*) is held inactive while the SSI is idle, and *SSIClk* transitions at the programmed frequency only during active transmission or reception of data. The idle state of *SSIClk* is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

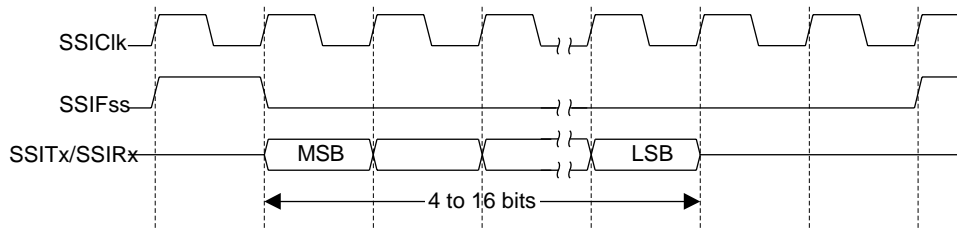
For Freescale SPI and MICROWIRE frame formats, the serial frame (*SSIFSS*) pin is active Low, and is asserted (pulled down) during the entire transmission of the frame.

For Texas Instruments synchronous serial frame format, the *SSIFSS* pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSI and the off-chip slave device drive their output data on the rising edge of *SSIClk*, and latch data from the other device on the falling edge.

Unlike the full-duplex transmission of the other two frame formats, the MICROWIRE format uses a special master-slave messaging technique, which operates at half-duplex. In this mode, when a frame begins, an 8-bit control message is transmitted to the off-chip slave. During this transmit, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the requested data. The returned data can be 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

14.2.4.1 Texas Instruments Synchronous Serial Frame Format

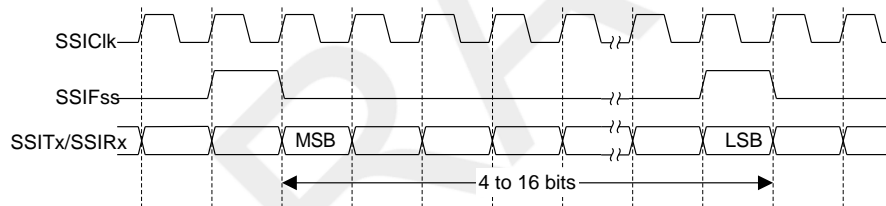
Figure 14-2 on page 328 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

Figure 14-2. TI Synchronous Serial Frame Format (Single Transfer)

In this mode, `SSIClk` and `SSIFss` are forced Low, and the transmit data line `SSITx` is tristated whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, `SSIFss` is pulsed High for one `SSIClk` period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of `SSIClk`, the MSB of the 4 to 16-bit data frame is shifted out on the `SSITx` pin. Likewise, the MSB of the received data is shifted onto the `SSIRx` pin by the off-chip serial slave device.

Both the SSI and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each `SSIClk`. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of `SSIClk` after the LSB has been latched.

Figure 14-3 on page 328 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

Figure 14-3. TI Synchronous Serial Frame Format (Continuous Transfer)

14.2.4.2 Freescale SPI Frame Format

The Freescale SPI interface is a four-wire interface where the `SSIFss` signal behaves as a slave select. The main feature of the Freescale SPI format is that the inactive state and phase of the `SSIClk` signal are programmable through the `SPO` and `SPH` bits within the `SSISCR0` control register.

SPO Clock Polarity Bit

When the `SPO` clock polarity control bit is Low, it produces a steady state Low value on the `SSIClk` pin. If the `SPO` bit is High, a steady state High value is placed on the `SSIClk` pin when data is not being transferred.

SPH Phase Control Bit

The `SPH` phase control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the `SPH` phase control bit is Low, data is captured on the first clock edge transition. If the `SPH` bit is High, data is captured on the second clock edge transition.

14.2.4.3 Freescale SPI Frame Format with SPO=0 and SPH=0

Single and continuous transmission signal sequences for Freescale SPI format with SPO=0 and SPH=0 are shown in Figure 14-4 on page 329 and Figure 14-5 on page 329.

Figure 14-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0

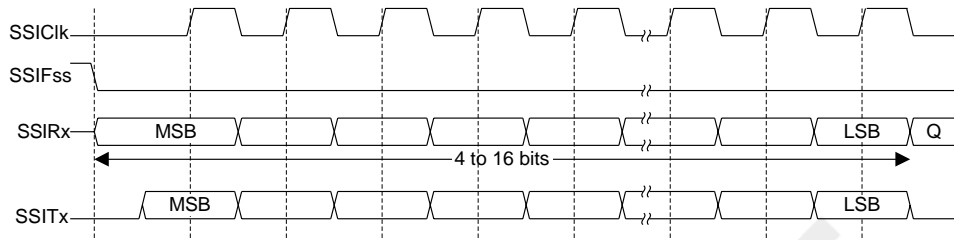
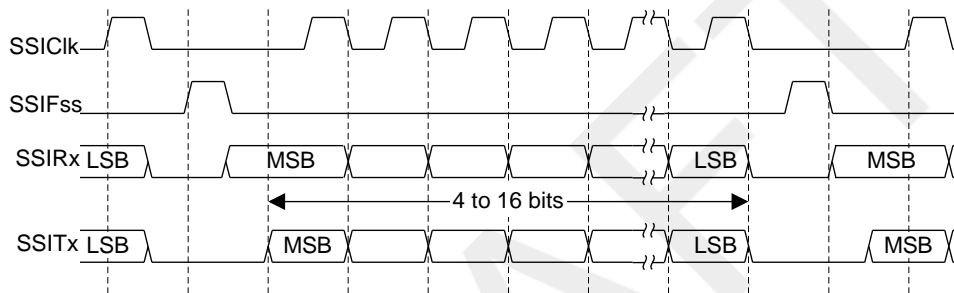


Figure 14-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0



Note: Q is undefined.

In this configuration, during idle periods:

- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. This causes slave data to be enabled onto the SSIRx input line of the master. The master SSITx output pad is enabled.

One half SSIClk period later, valid master data is transferred to the SSITx pin. Now that both the master and slave data have been set, the SSIClk master clock pin goes High after one further half SSIClk period.

The data is now captured on the rising and propagated on the falling edges of the SSIClk signal.

In the case of a single word transmission, after all bits of the data word have been transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

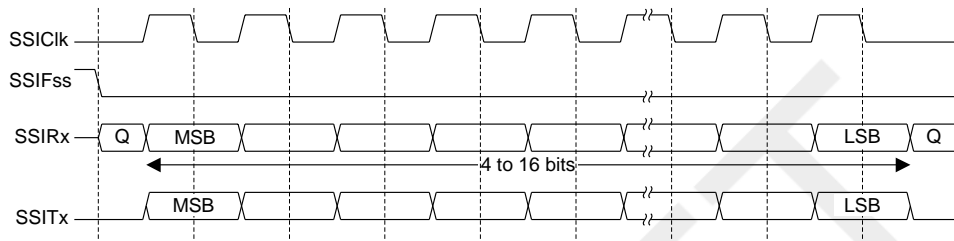
However, in the case of continuous back-to-back transmissions, the SSIFss signal must be pulsed High between each data word transfer. This is because the slave select pin freezes the data in its

serial peripheral register and does not allow it to be altered if the `SPH` bit is logic zero. Therefore, the master device must raise the `SSIFSS` pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the `SSIFSS` pin is returned to its idle state one `SSIClk` period after the last bit has been captured.

14.2.4.4 Freescale SPI Frame Format with `SPO=0` and `SPH=1`

The transfer signal sequence for Freescale SPI format with `SPO=0` and `SPH=1` is shown in Figure 14-6 on page 330, which covers both single and continuous transfers.

Figure 14-6. Freescale SPI Frame Format with `SPO=0` and `SPH=1`



Note: Q is undefined.

In this configuration, during idle periods:

- `SSIClk` is forced Low
- `SSIFss` is forced High
- The transmit data line `SSITx` is arbitrarily forced Low
- When the SSI is configured as a master, it enables the `SSIClk` pad
- When the SSI is configured as a slave, it disables the `SSIClk` pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the `SSIFss` master signal being driven Low. The master `SSITx` output is enabled. After a further one half `SSIClk` period, both master and slave valid data is enabled onto their respective transmission lines. At the same time, the `SSIClk` is enabled with a rising edge transition.

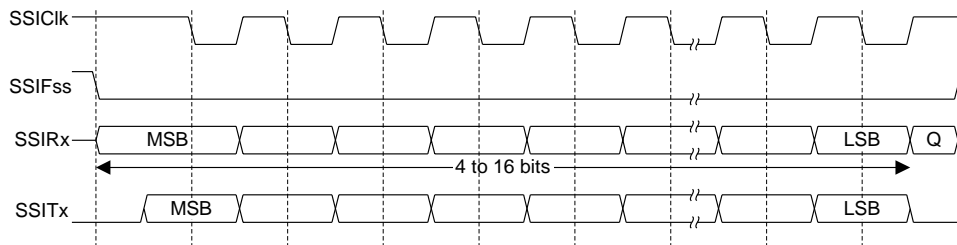
Data is then captured on the falling edges and propagated on the rising edges of the `SSIClk` signal.

In the case of a single word transfer, after all bits have been transferred, the `SSIFss` line is returned to its idle High state one `SSIClk` period after the last bit has been captured.

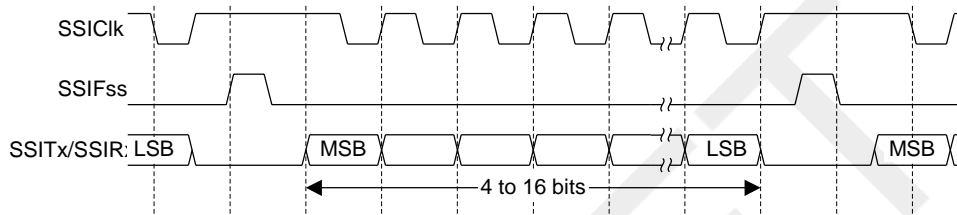
For continuous back-to-back transfers, the `SSIFss` pin is held Low between successive data words and termination is the same as that of the single word transfer.

14.2.4.5 Freescale SPI Frame Format with `SPO=1` and `SPH=0`

Single and continuous transmission signal sequences for Freescale SPI format with `SPO=1` and `SPH=0` are shown in Figure 14-7 on page 331 and Figure 14-8 on page 331.

Figure 14-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0

Note: Q is undefined.

Figure 14-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0

In this configuration, during idle periods:

- SSIClk is forced High
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low, which causes slave data to be immediately transferred onto the SSIRx line of the master. The master SSITx output pad is enabled.

One half period later, valid master data is transferred to the SSITx line. Now that both the master and slave data have been set, the SSIClk master clock pin becomes Low after one further half SSIClk period. This means that data is captured on the falling edges and propagated on the rising edges of the SSIClk signal.

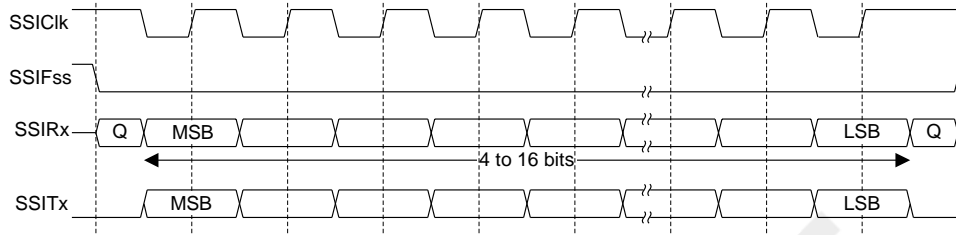
In the case of a single word transmission, after all bits of the data word are transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSIFss signal must be pulsed High between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is logic zero. Therefore, the master device must raise the SSIFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSIFss pin is returned to its idle state one SSIClk period after the last bit has been captured.

14.2.4.6 Freescale SPI Frame Format with SPO=1 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=1 and SPH=1 is shown in Figure 14-9 on page 332, which covers both single and continuous transfers.

Figure 14-9. Freescale SPI Frame Format with SPO=1 and SPH=1



Note: Q is undefined.

In this configuration, during idle periods:

- SSIClk is forced High
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. The master SSITx output pad is enabled. After a further one-half SSIClk period, both master and slave data are enabled onto their respective transmission lines. At the same time, SSIClk is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SSIClk signal.

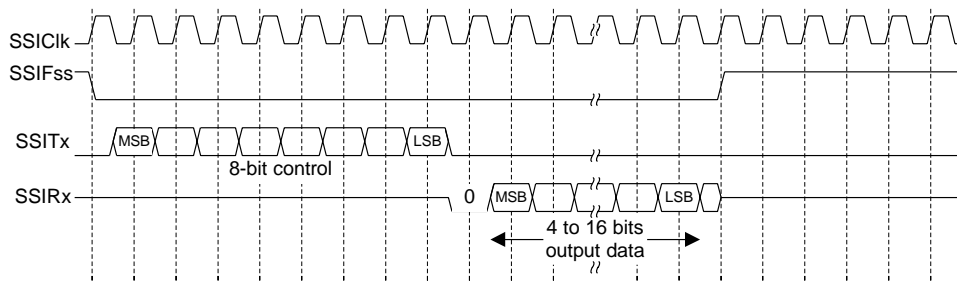
After all bits have been transferred, in the case of a single word transmission, the SSIFss line is returned to its idle high state one SSIClk period after the last bit has been captured.

For continuous back-to-back transmissions, the SSIFss pin remains in its active Low state, until the final bit of the last word has been captured, and then returns to its idle state as described above.

For continuous back-to-back transfers, the SSIFss pin is held Low between successive data words and termination is the same as that of the single word transfer.

14.2.4.7 MICROWIRE Frame Format

Figure 14-10 on page 333 shows the MICROWIRE frame format, again for a single frame. Figure 14-11 on page 334 shows the same format when back-to-back frames are transmitted.

Figure 14-10. MICROWIRE Frame Format (Single Frame)

MICROWIRE format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex, using a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSI to the off-chip slave device. During this transmission, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

In this configuration, during idle periods:

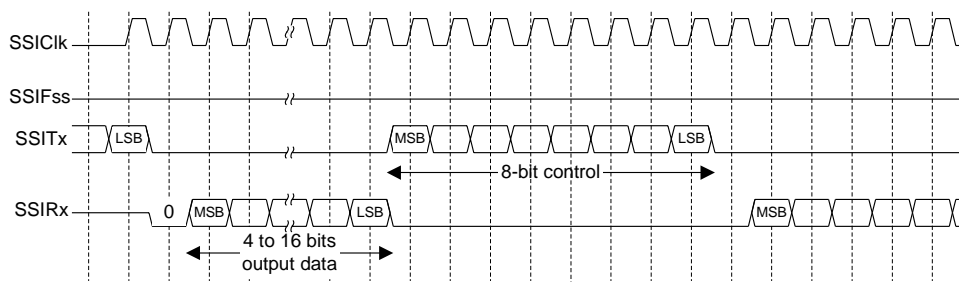
- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low

A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of SSIFss causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic, and the MSB of the 8-bit control frame to be shifted out onto the SSITx pin. SSIFss remains Low for the duration of the frame transmission. The SSIRx pin remains tristated during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on the rising edge of each SSIClk. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state, and the slave responds by transmitting data back to the SSI. Each bit is driven onto the SSIRx line on the falling edge of SSIClk. The SSI in turn latches each bit on the rising edge of SSIClk. At the end of the frame, for single transfers, the SSIFss signal is pulled High one clock period after the last bit has been latched in the receive serial shifter, which causes the data to be transferred to the receive FIFO.

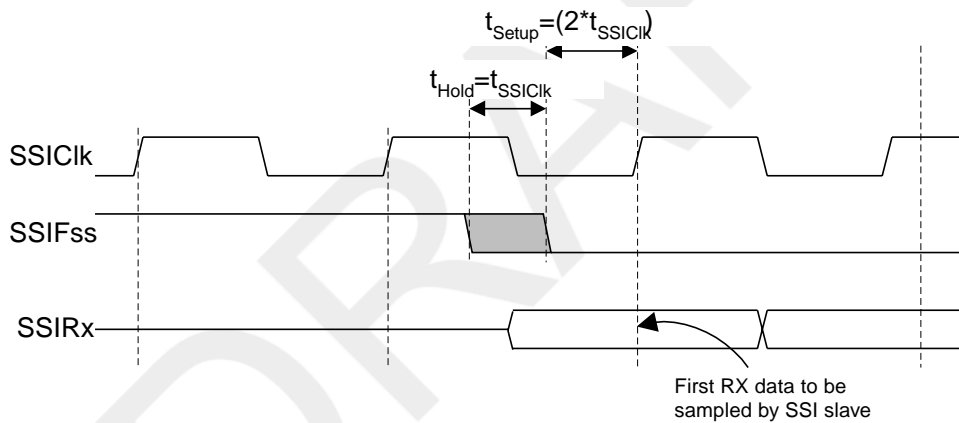
Note: The off-chip slave device can tristate the receive line either on the falling edge of SSIClk after the LSB has been latched by the receive shifter, or when the SSIFss pin goes High.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the SSIFss line is continuously asserted (held Low) and transmission of data occurs back-to-back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge of SSIClk, after the LSB of the frame has been latched into the SSI.

Figure 14-11. MICROWIRE Frame Format (Continuous Transfer)

In the MICROWIRE mode, the SSI slave samples the first bit of receive data on the rising edge of SSIClk after SSIFss has gone Low. Masters that drive a free-running SSIClk must ensure that the SSIFss signal has sufficient setup and hold margins with respect to the rising edge of SSIClk.

Figure 14-12 on page 334 illustrates these setup and hold time requirements. With respect to the SSIClk rising edge on which the first bit of receive data is to be sampled by the SSI slave, SSIFss must have a setup of at least two times the period of SSIClk on which the SSI operates. With respect to the SSIClk rising edge previous to this edge, SSIFss must have a hold of at least one SSIClk period.

Figure 14-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements

14.3 Initialization and Configuration

To use the SSI, its peripheral clock must be enabled by setting the SSI bit in the RCGC1 register.

For each of the frame formats, the SSI is configured using the following steps:

1. Ensure that the SSE bit in the SSICR1 register is disabled before making any configuration changes.
2. Select whether the SSI is a master or slave:
 - a. For master operations, set the SSICR1 register to 0x00000000.
 - b. For slave mode (output enabled), set the SSICR1 register to 0x00000004.
 - c. For slave mode (output disabled), set the SSICR1 register to 0x0000000C.
3. Configure the clock prescale divisor by writing the SSICPSR register.

4. Write the **SSICR0** register with the following configuration:
 - Serial clock rate (*SCR*)
 - Desired clock phase/polarity, if using Freescale SPI mode (*SPH* and *SPO*)
 - The protocol mode: Freescale SPI, TI SSF, MICROWIRE (*FRF*)
 - The data size (*DSS*)
5. Enable the SSI by setting the *SSE* bit in the **SSICR1** register.

As an example, assume the SSI must be configured to operate with the following parameters:

- Master operation
- Freescale SPI mode (*SPO*=1, *SPH*=1)
- 1 Mbps bit rate
- 8 data bits

Assuming the system clock is 20 MHz, the bit rate calculation would be:

$$F_{SSIClk} = F_{SysClk} / (CPSDVSR * (1 + SCR))$$

$$1 \times 10^6 = 20 \times 10^6 / (CPSDVSR * (1 + SCR))$$

In this case, if *CPSDVSR*=2, *SCR* must be 9.

The configuration sequence would be as follows:

1. Ensure that the *SSE* bit in the **SSICR1** register is disabled.
2. Write the **SSICR1** register with a value of 0x00000000.
3. Write the **SSICPSR** register with a value of 0x00000002.
4. Write the **SSICR0** register with a value of 0x000009C7.
5. The SSI is then enabled by setting the *SSE* bit in the **SSICR1** register to 1.

14.4 Register Map

Table 14-1 on page 335 lists the SSI registers. The offset listed is a hexadecimal increment to the register's address, relative to that SSI module's base address:

- SSI0: 0x4000.8000

Note: The SSI must be disabled (see the *SSE* bit in the **SSICR1** register) before any of the control registers are reprogrammed.

Table 14-1. SSI Register Map

Offset	Name	Type	Reset	Description	See page
0x000	SSICR0	R/W	0x0000.0000	SSI Control 0	337

Offset	Name	Type	Reset	Description	See page
0x004	SSICR1	R/W	0x0000.0000	SSI Control 1	339
0x008	SSIDR	R/W	0x0000.0000	SSI Data	340
0x00C	SSISR	RO	0x0000.0003	SSI Status	341
0x010	SSICPSR	R/W	0x0000.0000	SSI Clock Prescale	342
0x014	SSIIM	R/W	0x0000.0000	SSI Interrupt Mask	343
0x018	SSIRIS	RO	0x0000.0008	SSI Raw Interrupt Status	344
0x01C	SSIMIS	RO	0x0000.0000	SSI Masked Interrupt Status	345
0x020	SSIICR	W1C	0x0000.0000	SSI Interrupt Clear	346
0xFD0	SSIPeriphID4	RO	0x0000.0000	SSI Peripheral Identification 4	347
0xFD4	SSIPeriphID5	RO	0x0000.0000	SSI Peripheral Identification 5	348
0xFD8	SSIPeriphID6	RO	0x0000.0000	SSI Peripheral Identification 6	349
0xFDC	SSIPeriphID7	RO	0x0000.0000	SSI Peripheral Identification 7	350
0xFE0	SSIPeriphID0	RO	0x0000.0022	SSI Peripheral Identification 0	351
0xFE4	SSIPeriphID1	RO	0x0000.0000	SSI Peripheral Identification 1	352
0xFE8	SSIPeriphID2	RO	0x0000.0018	SSI Peripheral Identification 2	353
0xFEC	SSIPeriphID3	RO	0x0000.0001	SSI Peripheral Identification 3	354
0xFF0	SSIPCellID0	RO	0x0000.000D	SSI PrimeCell Identification 0	355
0xFF4	SSIPCellID1	RO	0x0000.00F0	SSI PrimeCell Identification 1	356
0xFF8	SSIPCellID2	RO	0x0000.0005	SSI PrimeCell Identification 2	357
0xFFC	SSIPCellID3	RO	0x0000.00B1	SSI PrimeCell Identification 3	358

14.5 Register Descriptions

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

Register 1: SSI Control 0 (SSICR0), offset 0x000

SSICR0 is control register 0 and contains bit fields that control various functions within the SSI module. Functionality such as protocol mode, clock rate and data size are configured in this register.

SSI Control 0 (SSICR0)

SSI0 base: 0x4000.8000
Offset 0x000
Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of 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	0	SSI Serial Clock Rate The value SCR is used to generate the transmit and receive bit rate of the SSI. The bit rate is: $BR = F_{SSIClk} / (CPSDVSR * (1 + SCR))$ where CPSDVSR is an even value from 2-254 programmed in the 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. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the SPH bit is 0, data is captured on the first clock edge transition. If SPH is 1, data is captured on the second clock edge transition.
6	SPO	R/W	0	SSI Serial Clock Polarity This bit is only applicable to the Freescale SPI Format. When the SPO bit is 0, it produces a steady state Low value on the SSIClk pin. If SPO is 1, a steady state High value is placed on the SSIClk pin when data is not being transferred.

Bit/Field	Name	Type	Reset	Description
5:4	FRF	R/W	0	SSI Frame Format Select The FRF values are defined as follows: FRF Value Frame Format 00 Freescale SPI Frame Format 01 Texas Instruments Synchronous Serial Frame Format 10 MICROWIRE Frame Format 11 Reserved
3:0	DSS	R/W	0	SSI Data Size Select The DSS values are defined as follows: DSS Value Data Size 0000-0010 Reserved 0011 4-bit data 0100 5-bit data 0101 6-bit data 0110 7-bit data 0111 8-bit data 1000 9-bit data 1001 10-bit data 1010 11-bit data 1011 12-bit data 1100 13-bit data 1101 14-bit data 1110 15-bit data 1111 16-bit data

Register 2: SSI Control 1 (SSICR1), offset 0x004

SSICR1 is control register 1 and contains bit fields that control various functions within the SSI module. Master and slave mode functionality is controlled by this register.

SSI Control 1 (SSICR1)

SSI0 base: 0x4000.8000
Offset 0x004
Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	SOD	R/W	0	SSI Slave Mode Output Disable This bit is relevant only in the Slave mode ($MS=1$). In multiple-slave systems, it is possible for the SSI master to broadcast a message to all slaves in the system while ensuring that only one slave drives data onto the serial output line. In such systems, the TXD lines from multiple slaves could be tied together. To operate in such a system, the SOD bit can be configured so that the SSI slave does not drive the $SSITx$ pin. 0: SSI can drive $SSITx$ output in Slave Output mode. 1: SSI must not drive the $SSITx$ output in Slave mode.
2	MS	R/W	0	SSI Master/Slave Select This bit selects Master or Slave mode and can be modified only when SSI is disabled ($SSE=0$). 0: Device configured as a master. 1: Device configured as a slave.
1	SSE	R/W	0	SSI Synchronous Serial Port Enable Setting this bit enables SSI operation. 0: SSI operation disabled. 1: SSI operation enabled. Note: This bit must be set to 0 before any control registers are reprogrammed.
0	LBM	R/W	0	SSI Loopback Mode Setting this bit enables Loopback Test mode. 0: Normal serial port operation enabled. 1: Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.

Register 3: SSI Data (SSIDR), offset 0x008

SSIDR is the data register and is 16-bits wide. When **SSIDR** is read, the entry in the receive FIFO (pointed to by the current FIFO read pointer) is accessed. As data values are removed by the SSI receive logic from the incoming data frame, they are placed into the entry in the receive FIFO (pointed to by the current FIFO write pointer).

When **SSIDR** is written to, the entry in the transmit FIFO (pointed to by the write pointer) is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. It is loaded into the transmit serial shifter, then serially shifted out onto the **SSITx** pin at the programmed bit rate.

When a data size of less than 16 bits is selected, the user must right-justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 bits is automatically right-justified in the receive buffer.

When the SSI is programmed for MICROWIRE frame format, the default size for transmit data is eight bits (the most significant byte is ignored). The receive data size is controlled by the programmer. The transmit FIFO and the receive FIFO are not cleared even when the **SSE** bit in the **SSICR1** register is set to zero. This allows the software to fill the transmit FIFO before enabling the SSI.

SSI Data (SSIDR)

SSI0 base: 0x4000.8000
Offset 0x008
Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of 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	0	SSI Receive/Transmit Data

A read operation reads the receive FIFO. A write operation writes the transmit FIFO.

Software must right-justify data when the SSI is programmed for a data size that is less than 16 bits. Unused bits at the top are ignored by the transmit logic. The receive logic automatically right-justifies the data.

Register 4: SSI Status (SSISR), offset 0x00C

SSISR is a status register that contains bits that indicate the FIFO fill status and the SSI busy status.

SSI Status (SSISR)

SSI0 base: 0x4000.8000

Offset 0x00C

Type RO, reset 0x0000.0003

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

Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0	Software should not rely on the value of 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 0: SSI is idle. 1: SSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.
3	RFF	RO	0	SSI Receive FIFO Full 0: Receive FIFO is not full. 1: Receive FIFO is full.
2	RNE	RO	0	SSI Receive FIFO Not Empty 0: Receive FIFO is empty. 1: Receive FIFO is not empty.
1	TNF	RO	1	SSI Transmit FIFO Not Full 0: Transmit FIFO is full. 1: Transmit FIFO is not full.
0	TFE	RO	1	SSI Transmit FIFO Empty 0: Transmit FIFO is not empty. 1: Transmit FIFO is empty.

Register 5: SSI Clock Prescale (SSICPSR), offset 0x010

SSICPSR is the clock prescale register and specifies the division factor by which the system clock must be internally divided before further use.

The value programmed into this register must be an even number between 2 and 254. The least-significant bit of the programmed number is hard-coded to zero. If an odd number is written to this register, data read back from this register has the least-significant bit as zero.

SSI Clock Prescale (SSICPSR)

SSI0 base: 0x4000.8000
 Offset 0x010
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0	Software should not rely on the value of a reserved 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	0	SSI Clock Prescale Divisor This value must be an even number from 2 to 254, depending on the frequency of SSI _{CLK} . The LSB always returns 0 on reads.

Register 6: SSI Interrupt Mask (SSIIM), offset 0x014

The **SSIIM** register is the interrupt mask set or clear register. It is a read/write register and all bits are cleared to 0 on reset.

On a read, this register gives the current value of the mask on the relevant interrupt. A write of 1 to the particular bit sets the mask, enabling the interrupt to be read. A write of 0 clears the corresponding mask.

SSI Interrupt Mask (SSIIM)

SSI0 base: 0x4000.8000
Offset 0x014
Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXIM	R/W	0	SSI Transmit FIFO Interrupt Mask 0: TX FIFO half-full or less condition interrupt is masked. 1: TX FIFO half-full or less condition interrupt is not masked.
2	RXIM	R/W	0	SSI Receive FIFO Interrupt Mask 0: RX FIFO half-full or more condition interrupt is masked. 1: RX FIFO half-full or more condition interrupt is not masked.
1	RTIM	R/W	0	SSI Receive Time-Out Interrupt Mask 0: RX FIFO time-out interrupt is masked. 1: RX FIFO time-out interrupt is not masked.
0	RORIM	R/W	0	SSI Receive Overrun Interrupt Mask 0: RX FIFO overrun interrupt is masked. 1: RX FIFO overrun interrupt is not masked.

Register 7: SSI Raw Interrupt Status (SSIRIS), offset 0x018

The **SSIRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

SSI Raw Interrupt Status (SSIRIS)

SSI0 base: 0x4000.8000

Offset 0x018

Type RO, reset 0x0000.0008

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXRIS	RO	1	SSI Transmit FIFO Raw Interrupt Status Indicates that the transmit FIFO is half full or less, when set.
2	RXRIS	RO	0	SSI Receive FIFO Raw Interrupt Status Indicates that the receive FIFO is half full or more, when set.
1	RTRIS	RO	0	SSI Receive Time-Out Raw Interrupt Status Indicates that the receive time-out has occurred, when set.
0	RORRIS	RO	0	SSI Receive Overrun Raw Interrupt Status Indicates that the receive FIFO has overflowed, when set.

Register 8: SSI Masked Interrupt Status (SSIMIS), offset 0x01C

The **SSIMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

SSI Masked Interrupt Status (SSIMIS)

SSI0 base: 0x4000.8000

Offset 0x01C

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXMIS	RO	0	SSI Transmit FIFO Masked Interrupt Status Indicates that the transmit FIFO is half full or less, when set.
2	RXMIS	RO	0	SSI Receive FIFO Masked Interrupt Status Indicates that the receive FIFO is half full or more, when set.
1	RTMIS	RO	0	SSI Receive Time-Out Masked Interrupt Status Indicates that the receive time-out has occurred, when set.
0	RORMIS	RO	0	SSI Receive Overrun Masked Interrupt Status Indicates that the receive FIFO has overflowed, when set.

Register 9: SSI Interrupt Clear (SSIICR), offset 0x020

The **SSIICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

SSI Interrupt Clear (SSIICR)

SSI0 base: 0x4000.8000
 Offset 0x020
 Type W1C, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of 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 0: No effect on interrupt. 1: Clears interrupt.
0	RORIC	W1C	0	SSI Receive Overrun Interrupt Clear 0: No effect on interrupt. 1: Clears interrupt.

Register 10: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0

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

SSI Peripheral Identification 4 (SSIPeriphID4)

SSI0 base: 0x4000.8000

Offset 0xFD0

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

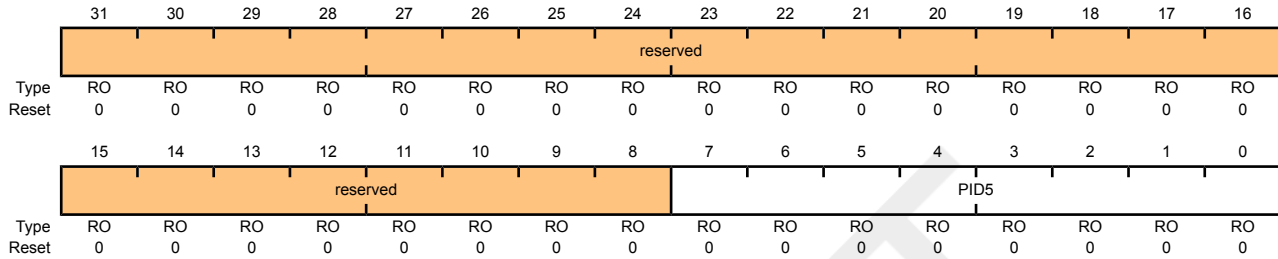
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	SSI Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 11: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4

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

SSI Peripheral Identification 5 (SSIPeriphID5)

SSI0 base: 0x4000.8000
 Offset 0xFD4
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	SSI Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

Register 12: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8

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

SSI Peripheral Identification 6 (SSIPeriphID6)

SSI0 base: 0x4000.8000

Offset 0xFD8

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID6							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

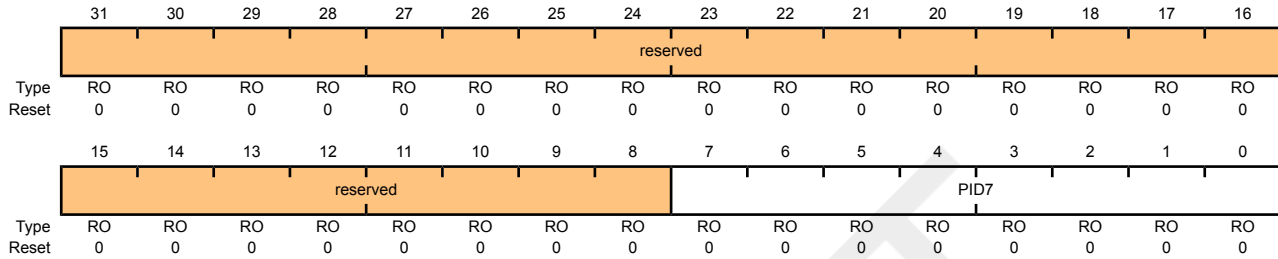
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	SSI Peripheral ID Register[23:16] Can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

Register 13: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC

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

SSI Peripheral Identification 7 (SSIPeriphID7)

SSI0 base: 0x4000.8000
 Offset 0xFDC
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	SSI Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

Register 14: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0

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

SSI Peripheral Identification 0 (SSIPeriphID0)

SSI0 base: 0x4000.8000

Offset 0xFE0

Type RO, reset 0x0000.0022

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1

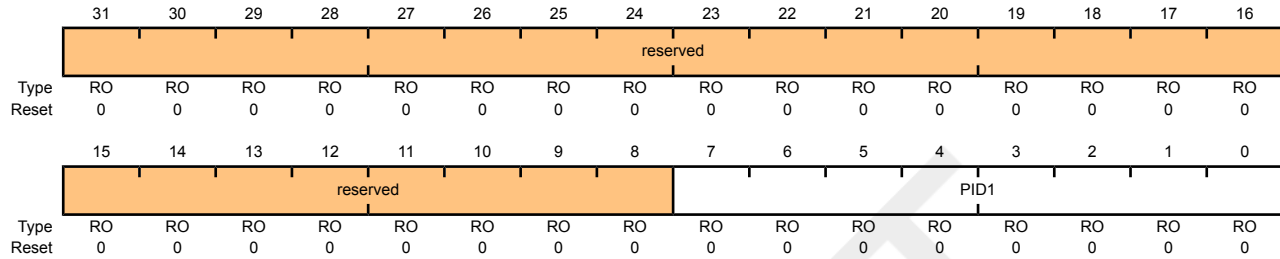
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x22	SSI Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 15: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4

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

SSI Peripheral Identification 1 (SSIPeriphID1)

SSI0 base: 0x4000.8000
 Offset 0xFE4
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	SSI Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 16: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8

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

SSI Peripheral Identification 2 (SSIPeriphID2)

SSI0 base: 0x4000.8000

Offset 0xFE8

Type RO, reset 0x0000.0018

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	SSI Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 17: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC

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

SSI Peripheral Identification 3 (SSIPeriphID3)

SSI0 base: 0x4000.8000
 Offset 0xFEC
 Type RO, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

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

Register 18: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0

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

SSI PrimeCell Identification 0 (SSIPCellID0)

SSI0 base: 0x4000.8000

Offset 0xFF0

Type RO, reset 0x0000.000D

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

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

Register 19: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4

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

SSI PrimeCell Identification 1 (SSIPCellID1)

SSI0 base: 0x4000.8000
 Offset 0xFF4
 Type RO, reset 0x0000.00F0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

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

Register 20: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8

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

SSI PrimeCell Identification 2 (SSIPCellID2)

SSI0 base: 0x4000.8000

Offset 0xFF8

Type RO, reset 0x0000.0005

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

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

Register 21: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC

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

SSI PrimeCell Identification 3 (SSIPCellID3)

SSI0 base: 0x4000.8000
 Offset 0xFFC
 Type RO, reset 0x0000.00B1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved 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

I²C

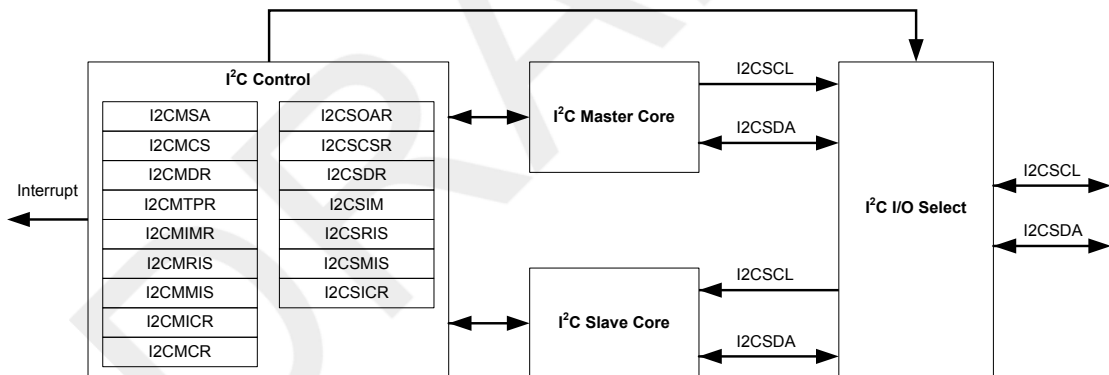
The Inter-Integrated Circuit (I²C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL), and interfaces to external I²C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I²C bus may also be used for system testing and diagnostic purposes in product development and manufacture. The LM3S8938 microcontroller includes two I²C modules, providing the ability to interact (both send and receive) with other I²C devices on the bus.

Devices on the I²C bus can be designated as either a master or a slave. Each Stellaris[®] I²C module supports both sending and receiving data as either a master or a slave, and also supports the simultaneous operation as both a master and a slave. There are a total of four I²C modes: Master Transmit, Master Receive, Slave Transmit, and Slave Receive. The Stellaris[®] I²C modules can operate at two speeds: Standard (100 Kbps) and Fast (400 Kbps).

Both the I²C master and slave can generate interrupts; the I²C master generates interrupts when a transmit or receive operation completes (or aborts due to an error) and the I²C slave generates interrupts when data has been sent or requested by a master.

15.1 Block Diagram

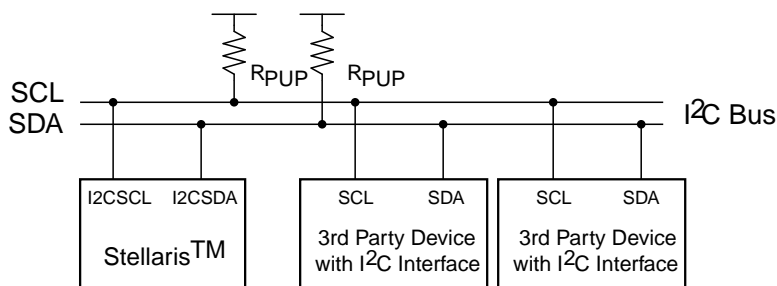
Figure 15-1. I²C Block Diagram



15.2 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 on page 360.

See "I²C" on page 515 for I²C timing diagrams.

Figure 15-2. I²C Bus Configuration

15.2.1 I²C Bus Functional Overview

The I²C bus uses only two signals: SDA and SCL, named I2CSDA and I2CSCL 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 360) is unrestricted, but each byte has to be followed by an acknowledge bit, and data must be transferred MSB first. When a receiver cannot receive another complete byte, it can hold the clock line SCL Low and force the transmitter into a wait state. The data transfer continues when the receiver releases the clock SCL.

15.2.1.1 START and STOP Conditions

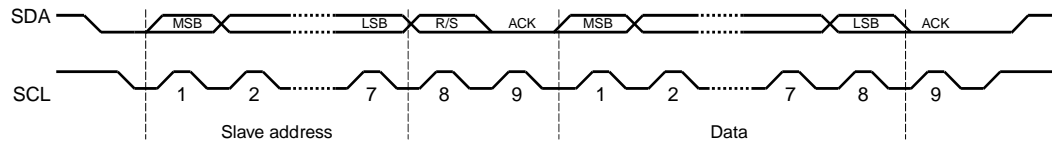
The protocol of the I²C bus defines two states to begin and end a transaction: START and STOP. A high-to-low transition on the SDA line while the SCL is high is defined as a START condition, and a low-to-high transition on the SDA line while SCL is high is defined as a STOP condition. The bus is considered busy after a START condition and free after a STOP condition. See Figure 15-3 on page 360.

Figure 15-3. START and STOP Conditions

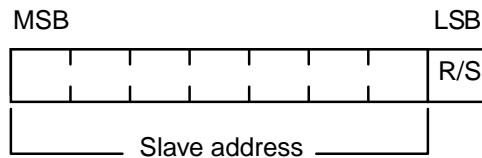


15.2.1.2 Data Format with 7-Bit Address

Data transfers follow the format shown in Figure 15-4 on page 361. After the START condition, a slave address is sent. This address is 7-bits long followed by an eighth bit, which is a data direction bit (R/S bit in the I2CMSA register). A zero indicates a transmit operation (send), and a one indicates a request for data (receive). A data transfer is always terminated by a STOP condition generated by the master, however, a master can initiate communications with another device on the bus by generating a repeated START condition and addressing another slave without first generating a STOP condition. Various combinations of receive/send formats are then possible within a single transfer.

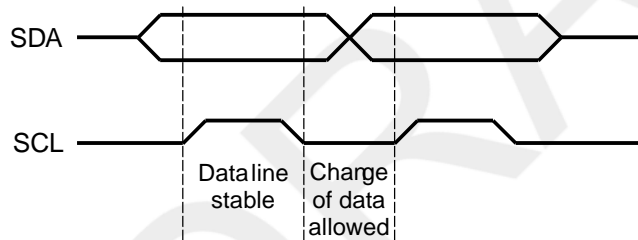
Figure 15-4. Complete Data Transfer with a 7-Bit Address

The first seven bits of the first byte make up the slave address (see Figure 15-5 on page 361). The eighth bit determines the direction of the message. A zero in the R/S position of the first byte means that the master will write (send) data to the selected slave, and a one in this position means that the master will receive data from the slave.

Figure 15-5. R/S Bit in First Byte

15.2.1.3 Data Validity

The data on the SDA line must be stable during the high period of the clock, and the data line can only change when SCL is low (see Figure 15-6 on page 361).

Figure 15-6. Data Validity During Bit Transfer on the I²C Bus

15.2.1.4 Acknowledge

All bus transactions have a required acknowledge clock cycle that is generated by the master. During the acknowledge cycle, the transmitter (which can be the master or slave) releases the SDA line. To acknowledge the transaction, the receiver must pull down SDA during the acknowledge clock cycle. The data sent out by the receiver during the acknowledge cycle must comply with the data validity requirements described in "Data Validity" on page 361.

When a slave receiver does not acknowledge the slave address, SDA must be left high by the slave so that the master can generate a STOP condition and abort the current transfer. If the master device is acting as a receiver during a transfer, it is responsible for acknowledging each transfer made by the slave. Since the master controls the number of bytes in the transfer, it signals the end of data to the slave transmitter by not generating an acknowledge on the last data byte. The slave transmitter must then release SDA to allow the master to generate the STOP or a repeated START condition.

15.2.1.5 Arbitration

A master may start a transfer only if the bus is idle. Its possible for two or more masters to generate a START condition within minimum hold time of the START condition. In these situations, an arbitration scheme takes place on the SDA line, while SCL is high. During arbitration, the first of the competing master devices to place a '1' (high) on SDA while another master transmits a '0' (low) will switch off its data output stage and retire until the bus is idle again.

Arbitration can take place over several bits. Its first stage is a comparison of address bits, and if both masters are trying to address the same device, arbitration continues on to the comparison of data bits.

15.2.2 Available Speed Modes

The 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 **I²C Master Timer Period (I2CMTPR)** register (see page 379).

The I²C clock period is calculated as follows:

$$SCL_PERIOD = 2 * (1 + TIMER_PRD) * (SCL_LP + SCL_HP) * CLK_PRD$$

For example:

CLK_PRD = 50 ns

TIMER_PRD = 2

SCL_LP=6

SCL_HP=4

yields a SCL frequency of:

$$1/T = 333 \text{ Khz}$$

Table 15-1 on page 362 gives examples of Timer period, system clock, and speed mode (Standard or Fast).

Table 15-1. 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
33Mhz	0x10	97.1 Kbps	0x04	330 Kbps
40Mhz	0x13	100 Kbps	0x04	400 Kbps

System Clock	Timer Period	Standard Mode	Timer Period	Fast Mode
50MHz	0x18	100 Kbps	0x06	357 Kbps

15.2.3 Interrupts

The I²C can generate interrupts when the following conditions are observed:

- Master transaction completed
- Master transaction error
- Slave transaction received
- Slave transaction requested

There is a separate interrupt signal for the I²C master and I²C modules. While both modules can generate interrupts for multiple conditions, only a single interrupt signal is sent to the interrupt controller.

15.2.3.1 I²C Master Interrupts

The I²C master module generates an interrupt when a transaction completes (either transmit or receive), or when an error occurs during a transaction. To enable the I²C master interrupt, software must write a '1' to the **I²C Master Interrupt Mask (I2CMIMR)** register. When an interrupt condition is met, software must check the **ERROR** bit in the **I²C Master Control/Status (I2CMCS)** register to verify that an error didn't occur during the last transaction. An error condition is asserted if the last transaction wasn't acknowledge by the slave or if the master was forced to give up ownership of the bus due to a lost arbitration round with another master. If an error is not detected, the application can proceed with the transfer. The interrupt is cleared by writing a '1' to the **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.

15.2.3.2 I²C Slave Interrupts

The slave module generates interrupts as it receives requests from an I²C master. To enable the I²C slave interrupt, write a '1' to 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 writing a '1' to 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.

15.2.4 Loopback Operation

The I²C modules can be placed into an internal loopback mode for diagnostic or debug work. This is accomplished 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.

15.2.5 Command Sequence Flow Charts

This section details the steps required to perform the various I²C transfer types in both master and slave mode.

15.2.5.1 I²C Master Command Sequences

The figures that follow show the command sequences available for the I²C master.

Figure 15-7. Master Single SEND

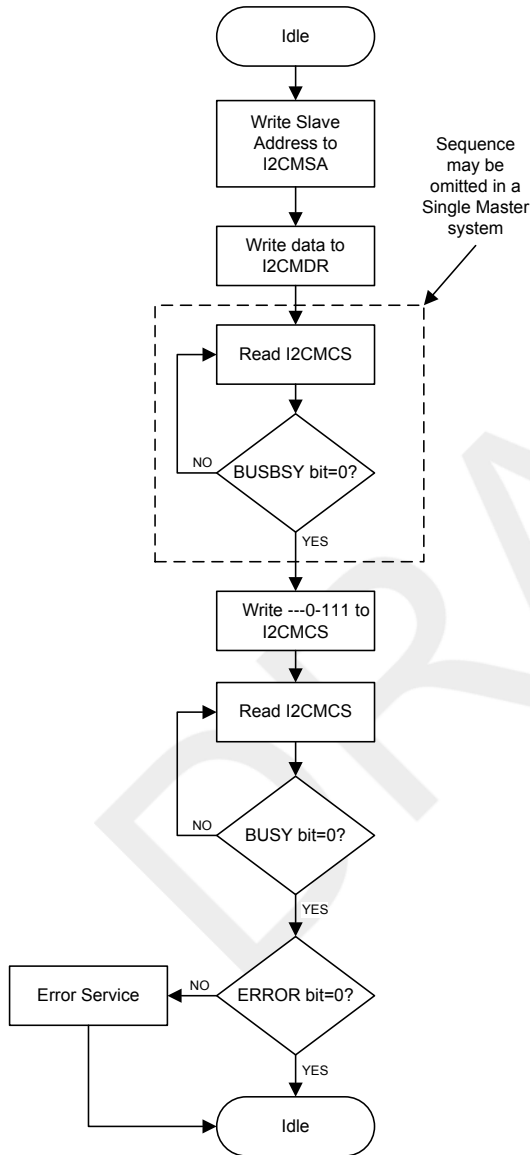


Figure 15-8. Master Single RECEIVE

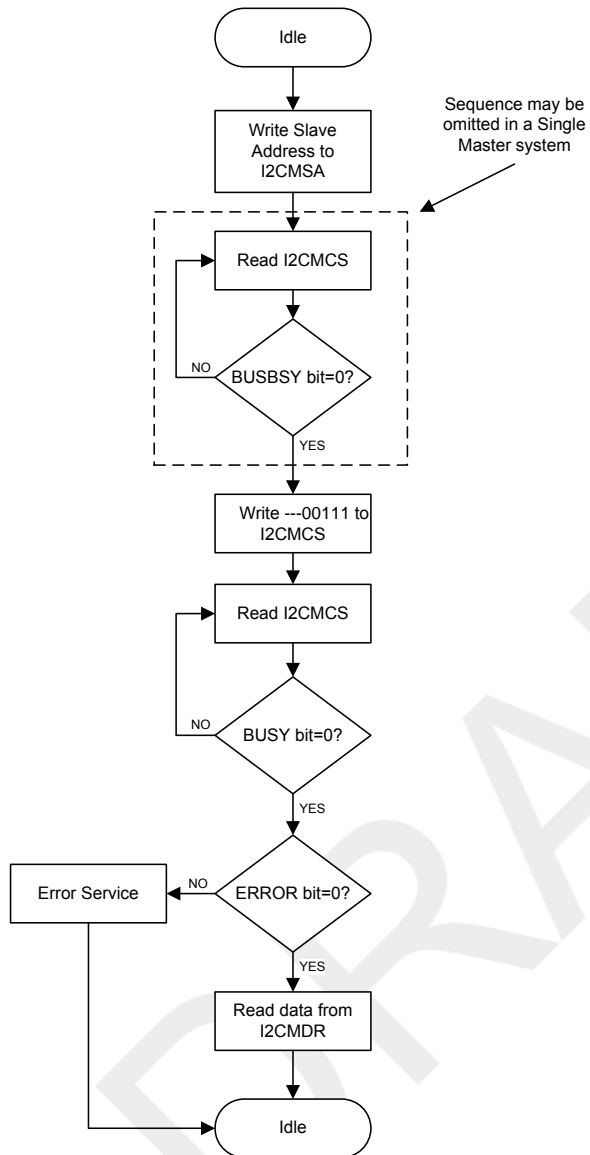


Figure 15-9. Master Burst SEND

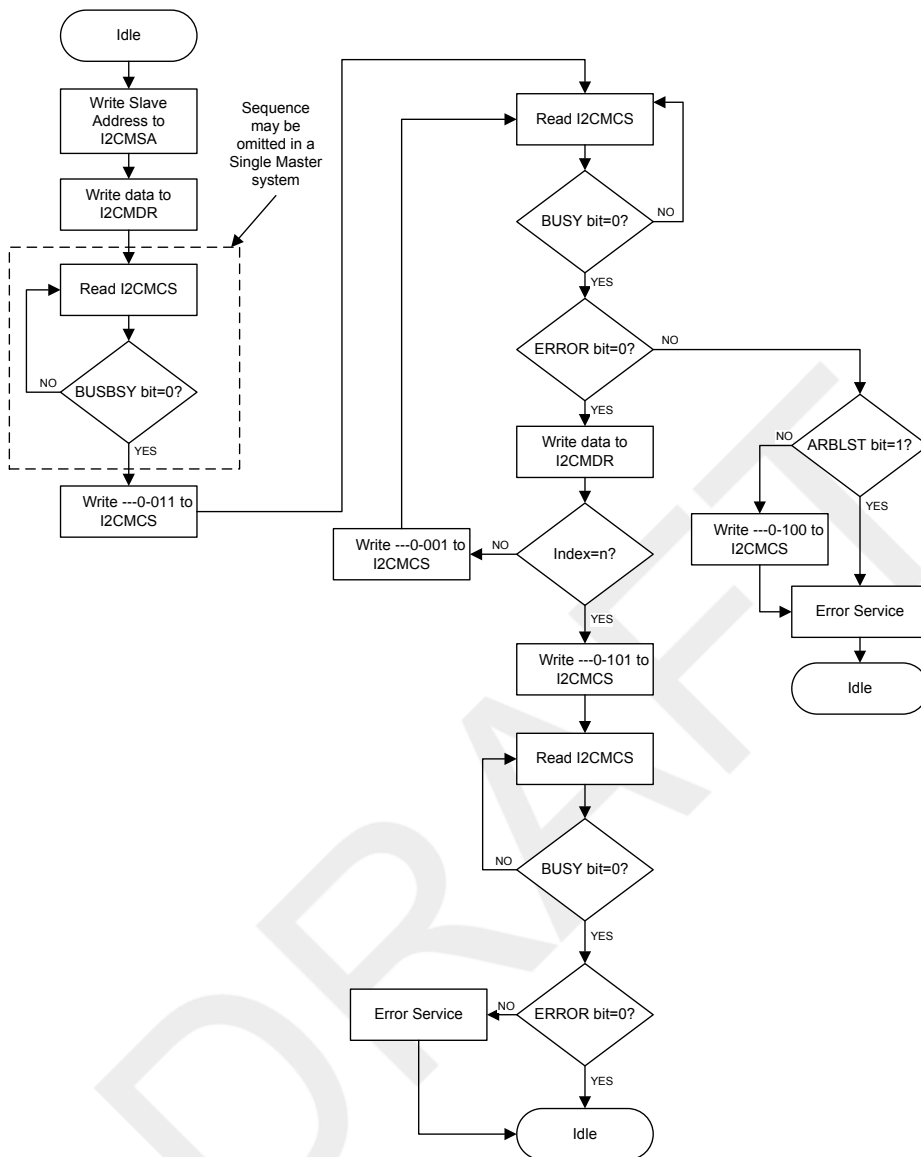


Figure 15-10. Master Burst RECEIVE

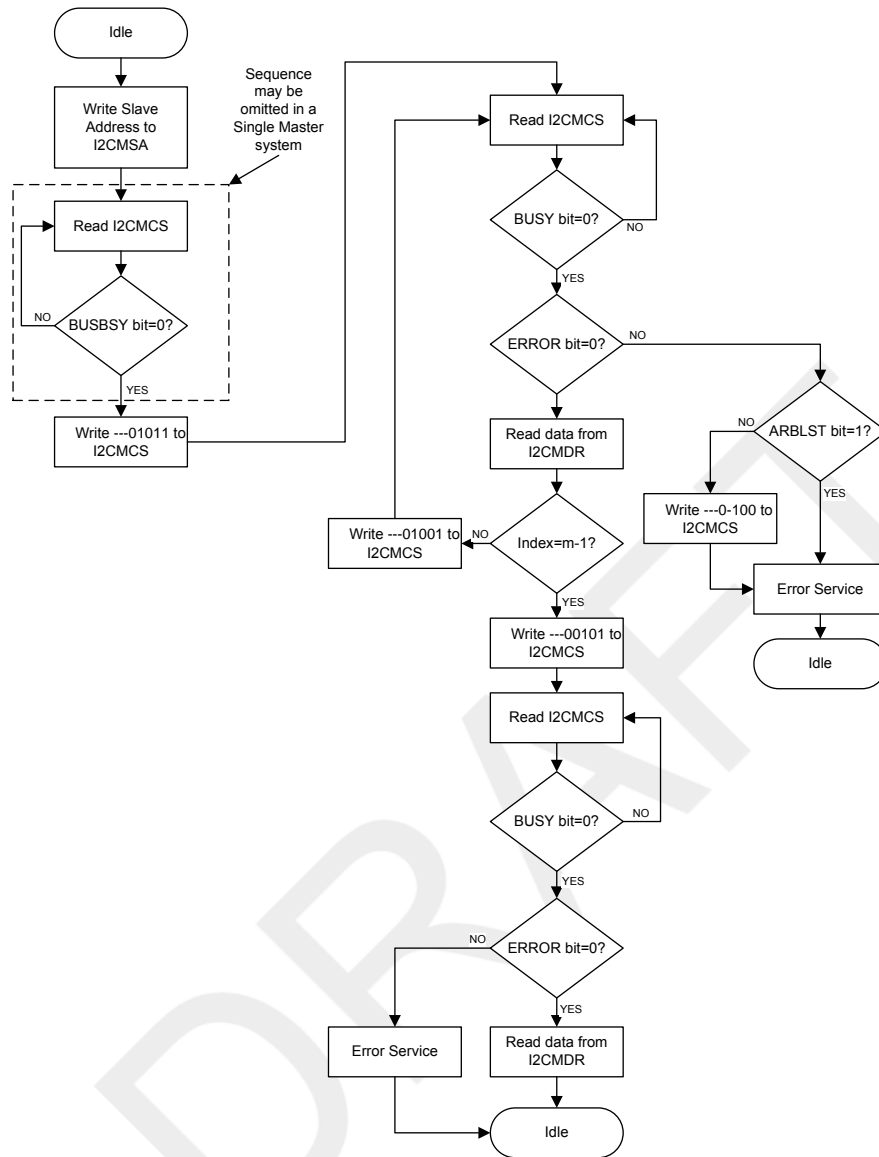


Figure 15-11. Master Burst RECEIVE after Burst SEND

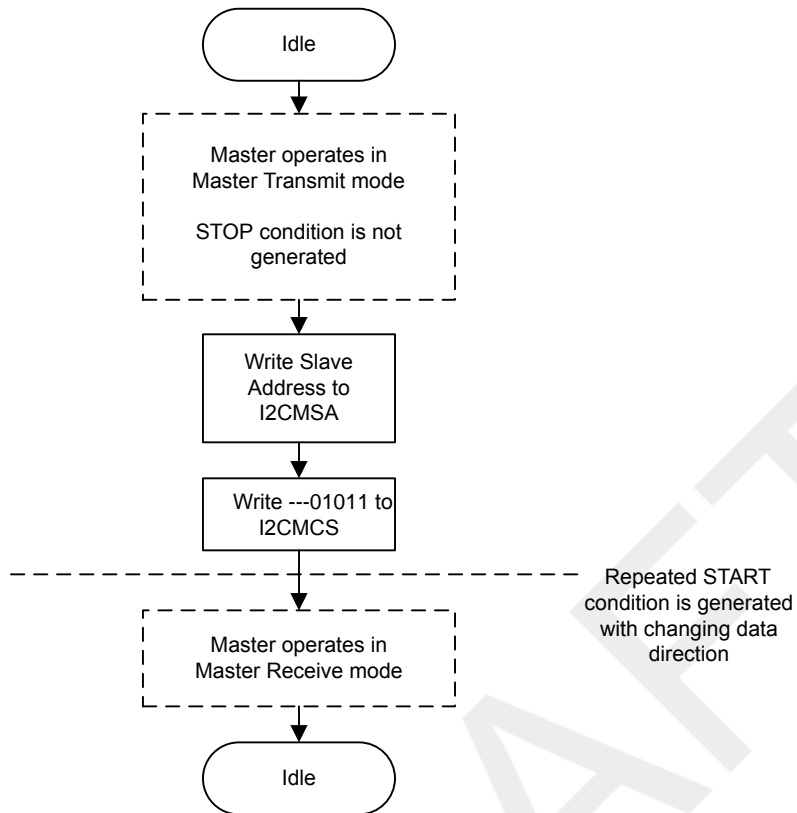
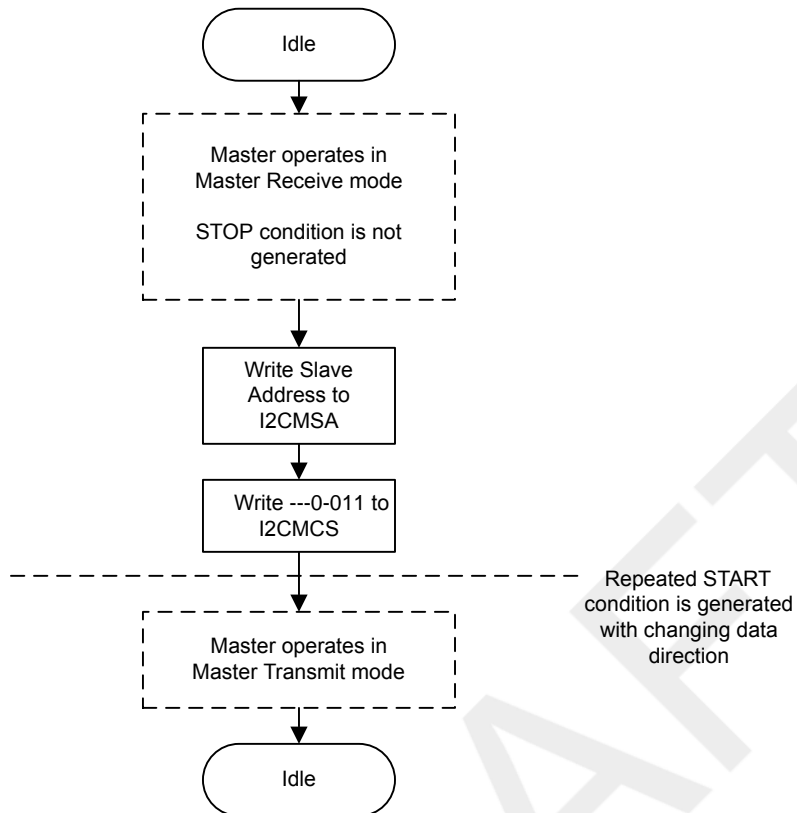


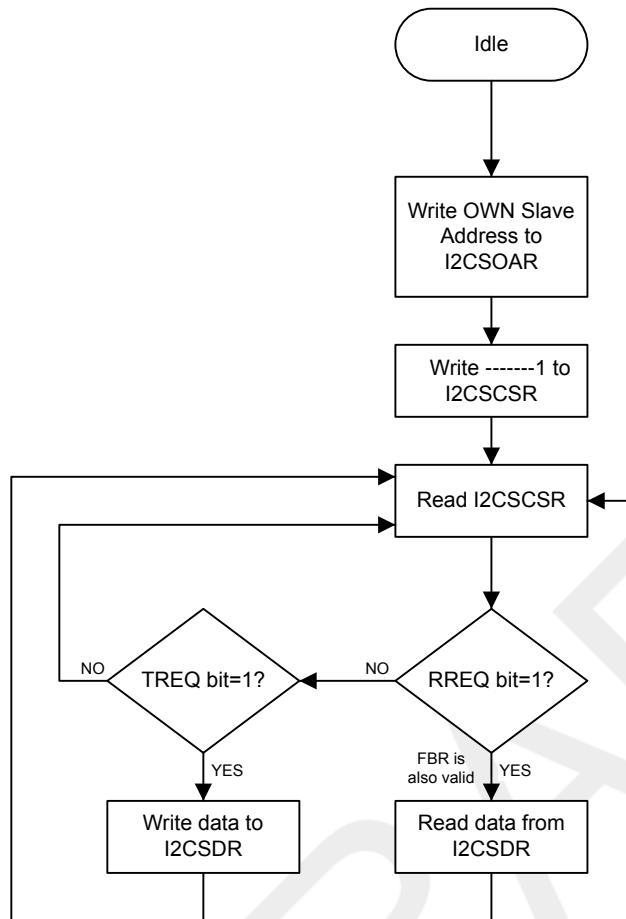
Figure 15-12. Master Burst SEND after Burst RECEIVE



15.2.5.2 I²C Slave Command Sequences

Figure 15-13 on page 370 presents the command sequence available for the I²C slave.

Figure 15-13. Slave Command Sequence



15.3 Initialization and Configuration

The following example shows how to configure the I²C module to send 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.
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module.
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register. Also, be sure to enable the same pins for Open Drain operation.
4. Initialize the I²C Master by writing the **I2CMCR** register with a value of 0x0000.0020.
5. Set the desired SCL clock speed of 100 Kbps by writing the **I2CMTPR** register with the correct value. The value written to the **I2CMTPR** register represents the number of system clock periods in one SCL clock period. The TPR value is determined by the following equation:

```

TPR = (System Clock / (2 * (SCL_LP + SCL_HP) * SCL_CLK)) - 1;
TPR = (20MHz / (2 * (6 + 4) * 100000)) - 1;
TPR = 9

```

Write the **I2CMTPR** register with the value of 0x0000.0009.

6. Specify the slave address of the master and that the next operation will be a Send by writing the **I2CMSA** register with a value of 0x0000.0076. This sets the slave address to 0x3B.
7. Place data (byte) to be sent in the data register by writing the **I2CMDR** register with the desired data.
8. Initiate a single byte send of the data from Master to Slave by writing the **I2CMCS** register with a value of 0x0000.0007 (STOP, START, RUN).
9. Wait until the transmission completes by polling the **I2CMCS** register's **BUSBSY** bit until it has been cleared.

15.4 I²C Register Map

Table 15-2 on page 371 lists the I²C registers. All addresses given are relative to the I²C base addresses for the master and slave:

- I²C Master 0: 0x4002.0000
- I²C Slave 0: 0x4002.0800
- I²C Master 1: 0x4002.1000
- I²C Slave 1: 0x4001.1800

Table 15-2. 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	373
0x004	I2CMCS	R/W	0x0000.0000	I2C Master Control/Status	374
0x008	I2CMDR	R/W	0x0000.0000	I2C Master Data	378
0x00C	I2CMTPR	R/W	0x0000.0001	I2C Master Timer Period	379
0x010	I2CMIMR	R/W	0x0000.0000	I2C Master Interrupt Mask	380
0x014	I2CMRIS	RO	0x0000.0000	I2C Master Raw Interrupt Status	381
0x018	I2CMMIS	RO	0x0000.0000	I2C Master Masked Interrupt Status	382
0x01C	I2CMICR	WO	0x0000.0000	I2C Master Interrupt Clear	383
0x020	I2CMCR	R/W	0x0000.0000	I2C Master Configuration	384
I²C Slave					
0x000	I2CSOAR	R/W	0x0000.0000	I2C Slave Own Address	386

Offset	Name	Type	Reset	Description	See page
0x004	I2CSCSR	RO	0x0000.0000	I2C Slave Control/Status	387
0x008	I2CSDR	R/W	0x0000.0000	I2C Slave Data	389
0x00C	I2CSIMR	R/W	0x0000.0000	I2C Slave Interrupt Mask	390
0x010	I2CSRIS	RO	0x0000.0000	I2C Slave Raw Interrupt Status	391
0x014	I2CSMIS	RO	0x0000.0000	I2C Slave Masked Interrupt Status	392
0x018	I2CSICR	WO	0x0000.0000	I2C Slave Interrupt Clear	393

15.5 Register Descriptions (I²C Master)

The remainder of this section lists and describes the I²C master registers, in numerical order by address offset. See also “Register Descriptions (I2C Slave)” on page 385.

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 Send (Low).

I2C Master Slave Address (I2CMSA)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x000

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:1	SA	R/W	0	I 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 Send (Low). 0: Send 1: Receive

Register 2: I²C Master Control/Status (I2CMCS), offset 0x004

This register accesses four control bits when written, and accesses seven status bits when read.

The status register consists of seven bits, which when read determine the state of the I²C bus controller.

The control register consists of four bits: the RUN, START, STOP, and ACK bits. The START bit causes the generation of the START, or REPEATED START condition.

The STOP bit determines if the cycle stops at the end of the data cycle, or continues on to a burst. To generate a single send cycle, the I²C Master Slave Address (I2CMSA) register is written with the desired address, the R/S bit is set to 0, and the Control register is written with ACK=X (0 or 1), STOP=1, START=1, and RUN=1 to perform the operation and stop. When the operation is completed (or aborted due an error), the interrupt pin becomes active and the data may be read from the I2CMDR register. When the I²C module operates in Master receiver mode, the ACK bit must be set normally to logic 1. This causes the I²C bus controller to send an acknowledge automatically after each byte. This bit must be reset when the I²C bus controller requires no further data to be sent from the slave transmitter.

Read-Only Status Register

I2C Master Control/Status (I2CMCS)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x004

Type 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										BUSBSY	IDLE	ARBLST	DATAACK	ADRACK	ERROR	BUSY
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R	R	R	R	R	R	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

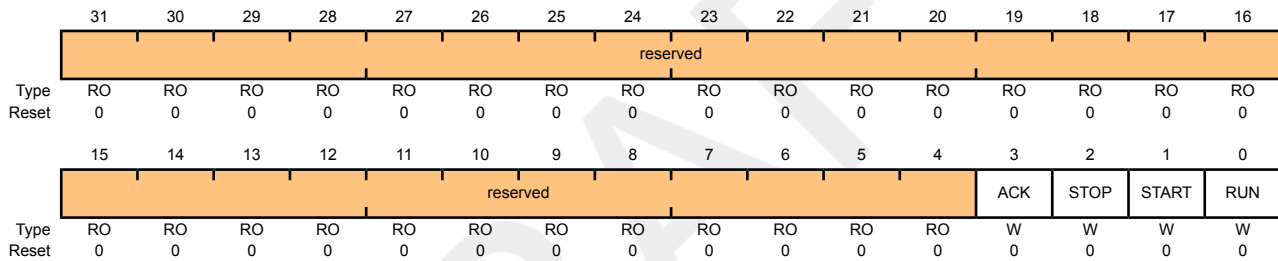
Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	BUSBSY	R	0	This bit specifies the state of the I ² C bus. If set, the bus is busy; otherwise, the bus is idle. The bit changes based on the START and STOP conditions.
5	IDLE	R	0	This bit specifies the I ² C controller state. If set, the controller is idle; otherwise the controller is not idle.
4	ARBLST	R	0	This bit specifies the result of bus arbitration. If set, the controller lost arbitration; otherwise, the controller won arbitration.
3	DATAACK	R	0	This bit specifies the result of the last data operation. If set, the transmitted data was not acknowledged; otherwise, the data was acknowledged.

Bit/Field	Name	Type	Reset	Description
2	ADRACK	R	0	This bit specifies the result of the last address operation. If set, the transmitted address was not acknowledged; otherwise, the address was acknowledged.
1	ERROR	R	0	This bit specifies the result of the last bus operation. If set, an error occurred on the last operation; otherwise, no error was detected. The error can be from the slave address not being acknowledged, the transmit data not being acknowledged, or because the controller lost arbitration.
0	BUSY	R	0	This bit specifies the state of the controller. If set, the controller is busy; otherwise, the controller is idle. When the BUSY bit is set, the other status bits are not valid.

Write-Only Control Register

I2C Master Control/Status (I2CMCS)

I2C Master 0 base: 0x4002.0000
 I2C Master 1 base: 0x4002.1000
 Offset 0x004
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ACK	W	0	When set, causes received data byte to be acknowledged automatically by the master. See field decoding in Table 15-3 on page 376.
2	STOP	W	0	When set, causes the generation of the STOP condition. See field decoding in Table 15-3 on page 376.
1	START	W	0	When set, causes the generation of a START or repeated START condition. See field decoding in Table 15-3 on page 376.
0	RUN	W	0	When set, allows the master to send or receive data. See field decoding in Table 15-3 on page 376.

Table 15-3. Write Field Decoding for I2CMCS[3:0] Field (Sheet 1 of 3)

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
	R/S	ACK	STOP	START	RUN	
Idle	0	X ^a	0	1	1	START condition followed by SEND (master goes to the Master Transmit state).
	0	X	1	1	1	START condition followed by a SEND and STOP condition (master remains in Idle state).
	1	0	0	1	1	START condition followed by RECEIVE operation with negative ACK (master goes to the Master Receive state).
	1	0	1	1	1	START condition followed by RECEIVE and STOP condition (master remains in Idle state).
	1	1	0	1	1	START condition followed by RECEIVE (master goes to the Master Receive state).
	1	1	1	1	1	Illegal.
	All other combinations not listed are non-operations.					
Master Transmit	X	X	0	0	1	SEND operation (master remains in Master Transmit state).
	X	X	1	0	0	STOP condition (master goes to Idle state).
	X	X	1	0	1	SEND followed by STOP condition (master goes to Idle state).
	0	X	0	1	1	Repeated START condition followed by a SEND (master remains in Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by SEND and STOP condition (master goes to Idle state).
	1	0	0	1	1	Repeated START condition followed by a RECEIVE operation with a negative ACK (master goes to Master Receive state).
	1	0	1	1	1	Repeated START condition followed by a SEND and STOP condition (master goes to Idle state).
	1	1	0	1	1	Repeated START condition followed by RECEIVE (master goes to Master Receive state).
	1	1	1	1	1	Illegal.
	All other combinations not listed are non-operations.					

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
	R/S	ACK	STOP	START	RUN	
Master Receive	X	0	0	0	1	RECEIVE operation with negative ACK (master remains in Master Receive state).
	X	X	1	0	0	STOP condition (master goes to Idle state). ^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 SEND (master goes to Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by SEND and STOP condition (master goes to Idle state).
All other combinations not listed are non-operations.						NOP.

a. An X in a table cell indicates the bit can be 0 or 1.

b. In Master Receive mode, a STOP condition should be generated only after a Data Negative Acknowledge executed by the master or an Address Negative Acknowledge executed by the slave.

Register 3: I²C Master Data (I2CMDR), offset 0x008

This register contains the data to be transmitted when in the Master Transmit state, and the data received when in the Master Receive state.

I2C Master Data (I2CMDR)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x008

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved 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 during transaction.

Register 4: I²C Master Timer Period (I2CMTPR), offset 0x00C

This register specifies the period of the SCL clock.

I2C Master Timer Period (I2CMTPR)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x00C

Type R/W, reset 0x0000.0001

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TPR	R/W	0x1	This field specifies the period of the SCL clock.

$$SCL_PRD = 2 * (1 + TPR) * (SCL_LP + SCL_HP) * CLK_PRD$$

where:

SCL_PRD is the SCL line period (I²C clock).

TPR is the Timer Period register value (range of 1 to 255).

SCL_LP is the SCL Low period (fixed at 6).

SCL_HP is the SCL High period (fixed at 4).

Register 5: 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 Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x010

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IM	R/W	0	This bit controls whether a raw interrupt is promoted to a controller interrupt. If set, the interrupt is not masked and the interrupt is promoted; otherwise, the interrupt is masked.

Register 6: I²C Master Raw Interrupt Status (I2CMRIS), offset 0x014

This register specifies whether an interrupt is pending.

I2C Master Raw Interrupt Status (I2CMRIS)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x014

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of 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	This bit specifies the raw interrupt state (prior to masking) of the I ² C master block. If set, an interrupt is pending; otherwise, an interrupt is not pending.

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 Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x018

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	MIS	RO	0	This bit specifies the raw interrupt state (after masking) of the I ² C master block. If set, an interrupt was signaled; otherwise, an interrupt has not been generated since the bit was last cleared.

Register 8: I²C Master Interrupt Clear (I2CMICR), offset 0x01C

This register clears the raw interrupt.

I2C Master Interrupt Clear (I2CMICR)

I2C Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x01C

Type WO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IC	WO	0	Interrupt Clear This bit controls the clearing of the raw interrupt. A write of 1 clears the interrupt; otherwise, a write of 0 has no affect on the interrupt state. A read of this register returns no meaningful data.

Register 9: 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 Master 0 base: 0x4002.0000

I2C Master 1 base: 0x4002.1000

Offset 0x020

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Software should not rely on the value of 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 This bit specifies whether the interface may operate in Slave mode. If set, Slave mode is enabled; otherwise, Slave mode is disabled.
4	MFE	R/W	0	I ² C Master Function Enable This bit specifies whether the interface may operate in Master mode. If set, Master mode is enabled; otherwise, Master mode is disabled and the interface clock is disabled.
3:1	reserved	RO	0	Software should not rely on the value of 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 This bit specifies whether the interface is operating normally or in Loopback mode. If set, the device is put in a test mode loopback configuration; otherwise, the device operates normally.

15.6 Register Descriptions (I2C Slave)

The remainder of this section lists and describes the I²C slave registers, in numerical order by address offset. See also "Register Descriptions (I²C Master)" on page 372.

DRAFT

Register 10: I²C Slave Own Address (I2CSOAR), offset 0x000

This register consists of seven address bits that identify the Stellaris[®] I²C device on the I²C bus.

I2C Slave Own Address (I2CSOAR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4001.1800

Offset 0x000

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of 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	0	I ² C Slave Own Address This field specifies bits A6 through A0 of the slave address.

Register 11: I²C Slave Control/Status (I2CCSR), offset 0x004

This register accesses one control bit when written, and three status bits when read.

The read-only Status register consists of three bits: the `FBR`, `RREQ`, and `TREQ` bits. The `First Byte Received (FBR)` bit is set only after the Stellaris[®] 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 (I2CCSR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4001.1800

Offset 0x004

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0	Software should not rely on the value of 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	Indicates that the first byte following the slave's own address is received. This bit is only valid when the <code>RREQ</code> bit is set, and is automatically cleared when data has been read from the I2CSDR register. Note: This bit is not used for slave transmit operations.
1	TREQ	RO	0	This bit specifies the state of the I ² C slave with regards to outstanding transmit requests. If set, the I ² C unit has been addressed as a slave transmitter and uses clock stretching to delay the master until data has been written to the I2CSDR register. Otherwise, there is no outstanding transmit request.
0	RREQ	RO	0	Receive Request This bit specifies the status of the I ² C slave with regards to outstanding receive requests. If set, the I ² C unit has outstanding receive data from the I ² C master and uses clock stretching to delay the master until the data has been read from the I2CSDR register. Otherwise, no receive data is outstanding.

Write-Only Control Register

I2C Slave Control/Status (I2CSCSR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4001.1800

Offset 0x004

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of 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 1=Enables the I ² C slave operation. 0=Disables the I ² C slave operation.

Register 12: I²C Slave Data (I2CSDR), offset 0x008

This register contains the data to be transmitted when in the Slave Transmit state, and the data received when in the Slave Receive state.

I2C Slave Data (I2CSDR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4001.1800

Offset 0x008

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x0	This field contains the data for transfer during a slave receive or transmit operation.

Register 13: I²C Slave Interrupt Mask (I2CSIMR), offset 0x00C

This register controls whether a raw interrupt is promoted to a controller interrupt.

I2C Slave Interrupt Mask (I2CSIMR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4001.1800

Offset 0x00C

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of 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	This bit controls whether a raw interrupt is promoted to a controller interrupt. If set, the interrupt is not masked and the interrupt is promoted; otherwise, the interrupt is masked.

Register 14: I²C Slave Raw Interrupt Status (I2CSRIS), offset 0x010

This register specifies whether an interrupt is pending.

I2C Slave Raw Interrupt Status (I2CSRIS)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4001.1800

Offset 0x010

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of 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	This bit specifies the raw interrupt state (prior to masking) of the I ² C slave block. If set, an interrupt is pending; otherwise, an interrupt is not pending.

Register 15: I²C Slave Masked Interrupt Status (I2CSMIS), offset 0x014

This register specifies whether an interrupt was signaled.

I2C Slave Masked Interrupt Status (I2CSMIS)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4001.1800

Offset 0x014

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of 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	This bit specifies the raw interrupt state (after masking) of the I ² C slave block. If set, an interrupt was signaled; otherwise, an interrupt has not been generated since the bit was last cleared.

Register 16: I²C Slave Interrupt Clear (I2CSICR), offset 0x018

This register clears the raw interrupt.

I2C Slave Interrupt Clear (I2CSICR)

I2C Slave 0 base: 0x4002.0800

I2C Slave 1 base: 0x4001.1800

Offset 0x018

Type WO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of 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	This bit controls the clearing of the raw interrupt. A write of 1 clears the interrupt; otherwise a write of 0 has no affect on the interrupt state. A read of this register returns no meaningful data.

16 Controller Area Network (CAN) Module

CAN

16.1 Controller Area Network Overview

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

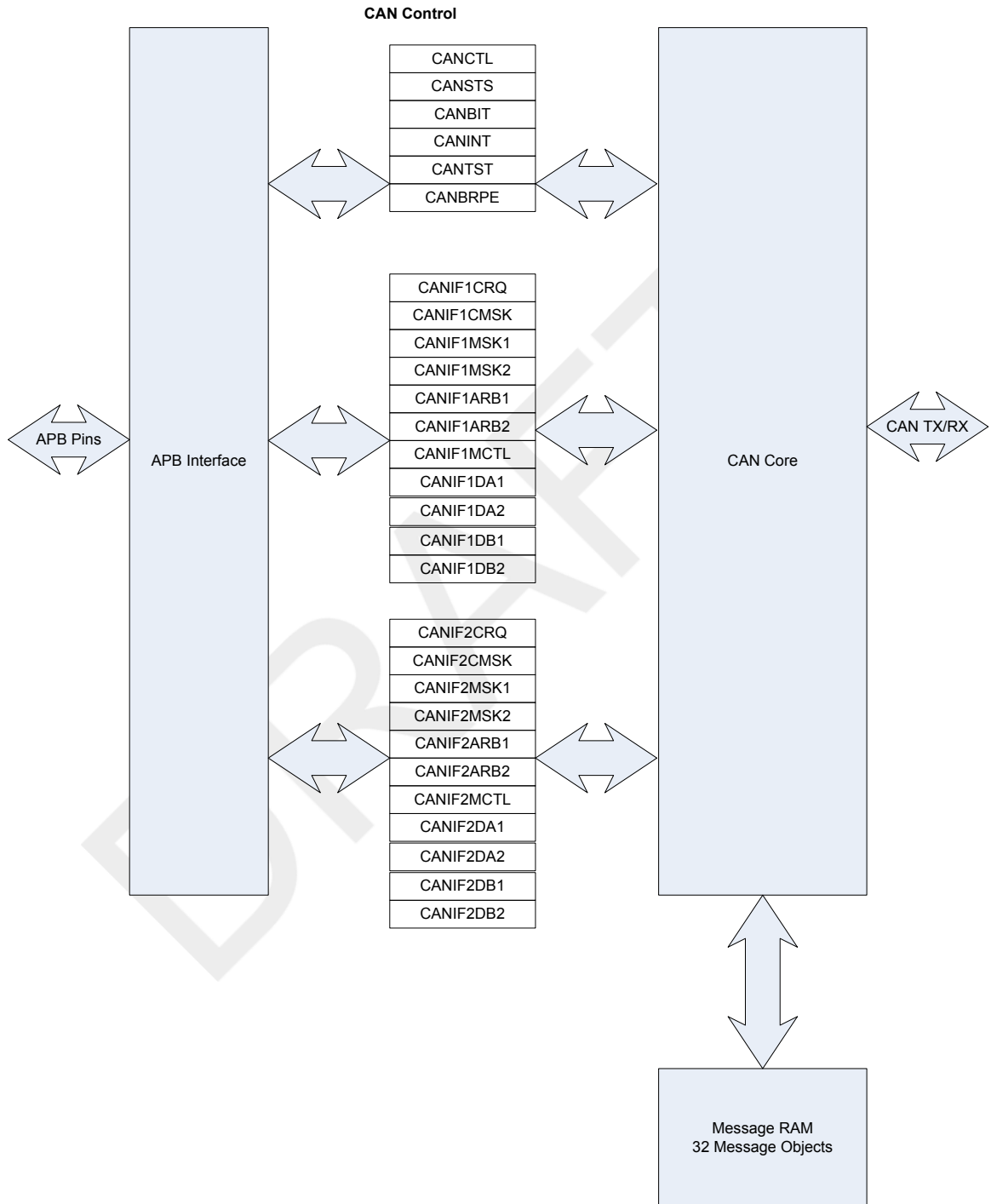
16.2 Controller Area Network Features

The Stellaris[®] CAN module supports the following features:

- CAN protocol version 2.0 part A/B.
- Bit rates up to 1 Mbps.
- 32 message objects.
- Each message object has its own identifier mask.
- Maskable interrupt.
- Disable Automatic Retransmission mode for Time Triggered CAN (TTCAN) applications.
- Programmable Loopback mode for self-test operation.
- Programmable FIFO mode.
- Gluelessly attach to an external CAN PHY through the CAN0Tx and CAN0Rx pins.

16.3 Controller Area Network Block Diagram

Figure 16-1. CAN Module Block Diagram



16.4 Controller Area Network Functional Description

The CAN module conforms to the CAN protocol version 2.0 (parts A and B). Message transfers that include data, remote, error, and overload frames with an 11-bit identifier (standard) or a 29-bit identifier (extended) are supported. Transfer rates can be programmed up to 1 Mbps.

The CAN module consists of three major parts:

- CAN protocol controller and message handler
- Message memory
- CAN register interface

The protocol controller transfers and receives the serial data from the CAN bus and passes the data on to the message handler. The message handler then loads this information into the appropriate message object based on the current filtering and identifiers in the message object memory. The message handler is also responsible for generating interrupts based on events on the CAN bus.

The message object memory is a set of 32 identical memory blocks that hold the current configuration, status, and actual data for each message object. These are accessed via the CAN message object register interface. The message memory is not directly accessible in the Stellaris memory map, so the Stellaris[®] CAN controller provides an interface to communicate with the message memory.

The CAN message object register interface provides two register sets for communicating with the message objects. Since there is no direct access to the message object memory, these two interfaces must be used to read or write to each message object. The two message object interfaces allow parallel access to the CAN controller message objects when multiple objects may have new information that needs to be processed.

16.4.1 Initialization

The software initialization is started by setting the `INIT` bit in the **CAN Control (CANCTL)** register, with software or by a hardware reset, or by going bus-off, which occurs when the transmitter's error counter exceeds a count of 255. While `INIT` is set, all message transfers to and from the CAN bus are stopped and the status of the CAN transmit output is recessive (High). Entering the initialization state does not change the configuration of the CAN controller, the message objects or the error counters. However, some configuration registers are only accessible when in the initialization state.

To initialize the CAN controller, set the **CAN Bit Timing (CANBIT)** register and configure each message object. If a message object is not needed, it is sufficient to set it as not valid by clearing the `MsgVal` bit in the **CANIFnARB2** register. Otherwise, the whole message object has to be initialized, as the fields of the message object may not have valid information causing unexpected results. Access to the **CAN Bit Timing (CANBIT)** register and to the **CAN Baud Rate Prescaler Extension (CANBRPE)** register to configure the bit timing are enabled when both the `INIT` and `CCE` bits in the **CANCTL** register are set. To leave the initialization state, the `INIT` bit must be cleared. Afterwards, the internal Bit Stream Processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus Idle) before it takes part in bus activities and starts message transfers. The initialization of the message objects is independent of being in the initialization state and can be done on the fly, but message objects should all be configured to particular identifiers or set to not valid before the BSP starts the message transfer. To change the configuration of a message object during normal operation, set the `MsgVal` bit in the **CANIFnARB2** register to 0 (not valid). When the configuration is completed, `MsgVal` is set to 1 again (valid).

16.4.2 Operation

Once the CAN module is initialized and the `INIT` bit in the **CANCTL** register is reset to 0, the CAN module synchronizes itself to the CAN bus and starts the message transfer. As messages are received, they are stored in their appropriate message objects if they pass the message handler's filtering. The whole message (including all arbitration bits, data-length code, and eight data bytes) is stored in the message object. If the Identifier Mask (the `MSK` bits in the **CANIFnMSKn** registers) is used, the arbitration bits which are masked to "don't care" may be overwritten in the message object.

The CPU may read or write each message any time via the CAN Interface Registers (**CANIFnCRQ**, **CANIFnCMSK**, **CANIFnMSKn**, **CANIFnARBn**, **CANIFnMCTL**, **CANIFnDAn**, and **CANIFnDBn**). The message handler guarantees data consistency in case of concurrent accesses.

The transmission of message objects are under the control of the software that is managing the CAN hardware. These can be message objects used for one-time data transfers, or permanent message objects used to respond in a more periodic manner. Permanent message objects have all arbitration and control set up and only the data bytes are updated. To start the transmission, the `TxRqst` bit in the **CANTXRQn** register and the `NewDat` bit in the **CANNWDAn** register are set. If several transmit messages are assigned to the same message object (when the number of message objects is not sufficient), the whole message object has to be configured before the transmission of this message is requested.

The transmission of any number of message objects may be requested at the same time; they are transmitted according to their internal priority, which is based on the message identifier for the message object. Messages may be updated or set to not valid any time, even when their requested transmission is still pending. The old data is discarded when a message is updated before its pending transmission has started. Depending on the configuration of the message object, the transmission of a message may be requested autonomously by the reception of a remote frame with a matching identifier.

There are two sets of CAN Interface Registers (**CANIF1x** and **CANIF2x**) which are used to access the Message Objects in the Message RAM. The CAN controller coordinates transfers to and from the Message RAM to and from the registers. The function of the two sets are independent and identical and can be used to queue transactions.

16.4.3 Transmitting Message Objects

If the internal transmit shift register of the CAN module is ready for loading, and if there is no data transfer between the CAN Interface Registers and message RAM, the valid message object with the highest priority and which has a pending transmission request is loaded into the transmit shift register by the message handler and the transmission is started. The message object's `NewDat` bit is reset and can be viewed in the **CANNWDAn** register. After a successful transmission, and if no new data was written to the message object since the start of the transmission, the `TxRqst` bit in the **CANIFnCMSK** register is reset. If the `TxIE` bit in the **CANIFnMCTL** register is set, the `IntPnd` bit in the **CANIFnMCTL** register is set after a successful transmission. If the CAN module has lost the arbitration or if an error occurred during the transmission, the message is re-transmitted as soon as the CAN bus is free again. If, meanwhile, the transmission of a message with higher priority has been requested, the messages are transmitted in the order of their priority.

16.4.4 Configuring a Transmit Message Object

Table 16-1 on page 398 specifies the bit settings for a transmit message object.

Table 16-1. Transmit Message Object Bit Settings

Register	CANIFnARB2	CANIFnCMSK			CANIFnMCTL	CANIFnARB2	CANIFnMCTL						
		Arb	Data	Mask			EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd
Value	1	appl	appl	appl	1	1	0	0	0	appl	0	appl	0

The `Xtd` and `ID` bit fields in the **CANIFnARBn** registers are set by an application. They define the identifier and type of the outgoing message. If an 11-bit Identifier (Standard Frame) is used, it is programmed to bits [28:18] of **CANIFnARB1**, as bits 17:0 of **CANIFnARBn** are not used by the CAN controller for 11-bit identifiers.

If the `TxIE` bit is set, the `IntPnd` bit is set after a successful transmission of the message object.

If the `RmtEn` bit is set, a matching received Remote Frame causes the `TxRqst` bit to be set and the Remote Frame is autonomously answered by a Data Frame with the data from the message object.

The `DLC` bit in the **CANIFnMCTL** register is set by an application. `TxRqst` and `RmtEn` may not be set before the data is valid.

The CAN mask registers (`Msk` bits in **CANIFnMSKn**, `UMask` bit in **CANIFnMCTL** register, and `MXtd` and `MDir` bits in **CANIFnMSK2** register) may be used (`UMask=1`) to allow groups of Remote Frames with similar identifiers to set the `TxRqst` bit. The `Dir` bit should not be masked.

16.4.5 Updating a Transmit Message Object

The CPU may update the data bytes of a Transmit Message Object any time via the CAN Interface Registers and neither the `MsgVal` nor the `TxRqst` bits have to be reset before the update.

Even if only a part of the data bytes are to be updated, all four bytes of the corresponding **CANIFnDAn** or **CANIFnDBn** register have to be valid before the content of that register is transferred to the message object. Either the CPU has to write all four bytes into the **CANIFnDAn** or **CANIFnDBn** register or the message object is transferred to the **CANIFnDAn** or **CANIFnDBn** register before the CPU writes the new data bytes.

In order to just update the data in a message object, the `WR`, `NewDat`, `DataA`, and `DataB` bits are written to the **CAN IFn Command Mask (CANIFnMSKn)** register, followed by writing the **CAN IFn Data** registers, and then the number of the message object is written to the **CAN IFn Command Request (CANIFnCRQ)** register, to update the data bytes and the `TxRqst` bit at the same time.

To prevent the reset of `TxRqst` at the end of a transmission that may already be in progress while the data is updated, `NewDat` has to be set together with `TxRqst`. When `NewDat` is set together with `TxRqst`, `NewDat` is reset as soon as the new transmission has started.

16.4.6 Accepting Received Message Objects

When the arbitration and control field (`ID + Xtd + RmtEn + DLC`) of an incoming message is completely shifted into the CAN module, the message handling capability of the module starts scanning the message RAM for a matching valid message object. To scan the message RAM for a matching message object, the Acceptance Filtering unit is loaded with the arbitration bits from the core. Then the arbitration and mask fields (including `MsgVal`, `UMask`, `NewDat`, and `EoB`) of message object 1 are loaded into the Acceptance Filtering unit and compared with the arbitration field from the shift register. This is repeated with each following message object until a matching message object is found or until the end of the message RAM is reached. If a match occurs, the scanning is stopped and the message handler proceeds depending on the type of frame received.

16.4.7 Receiving a Data Frame

The message handler stores the message from the CAN module receive shift register into the respective message object in the message RAM. It stores the data bytes, all arbitration bits, and the Data Length Code into the corresponding message object. This is implemented to keep the data bytes connected with the identifier even if arbitration mask registers are used. The `CANIFnMCTL.NewDat` bit is set to indicate that new data has been received. The CPU should reset `CANIFnMCTL.NewDat` when it reads the message object to indicate to the controller that the message has been received and the buffer is free to receive more messages. If the CAN controller receives a message and the `CANIFnMCTL.NewDat` bit was already set, the `MsgLst` bit is set to indicate that the previous data was lost. If the `CANIFnMCTL.RxIE` bit is set, the `CANIFnMCTL.IntPnd` bit is set, causing the **CANINT** interrupt register to point to the message object that just received a message. The `CANIFnMCTL.TxRqst` bit of this message object is reset to prevent the transmission of a Remote Frame, while the requested Data Frame has just been received.

16.4.8 Receiving a Remote Frame

When a Remote Frame is received, three different configurations of the matching message object have to be considered:

- `Dir = 1` (direction = transmit), `RmtEn = 1`, `UMask = 1` or `0`

At the reception of a matching Remote Frame, the `TxRqst` bit of this message object is set. The rest of the message object remains unchanged.

- `Dir = 1` (direction = transmit), `RmtEn = 0`, `UMask = 0`

At the reception of a matching Remote Frame, the `TxRqst` bit of this message object remains unchanged; the Remote Frame is ignored. This remote frame is disabled and will not automatically respond or indicate that the remote frame ever happened.

- `Dir = 1` (direction = transmit), `RmtEn = 0`, `UMask = 1`

At the reception of a matching Remote Frame, the `TxRqst` bit of this message object is reset. The arbitration and control field (`ID + Xtd + RmtEn + DLC`) from the shift register is stored into the message object in the message RAM and the `NewDat` bit of this message object is set. The data field of the message object remains unchanged; the Remote Frame is treated similar to a received Data Frame. This is useful for a remote data request from another CAN device for which the Stellaris® controller does not have readily available data. The software must fill the data and answer the frame manually.

16.4.9 Receive/Transmit Priority

The receive/transmit priority for the message objects is controlled by the message number. Message object 1 has the highest priority, while message object 32 has the lowest priority. If more than one transmission request is pending, the message objects are transmitted in order based on the message object with the lowest message number. This should not be confused with the message identifier as that priority is enforced by the CAN bus. This means that if message object 1 and message object 2 both have valid messages that need to be transmitted, message object 1 will always be transmitted first regardless of the message identifier in the message object itself.

16.4.10 Configuring a Receive Message Object

Table 16-2 on page 400 specifies the bit settings for a transmit message object.

Table 16-2. Receive Message Object Bit Settings

Register	CANIFnARB2	CANIFnCMSK			CANIFnMCTL	CANIFnARB2	CANIFnMCTL						
		MsgVal	Arb	Data			Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE
Value	1	appl	appl	appl	1	0	0	0	appl	0	0	0	0

The CAN arbitration registers ($ID[28:0]$ and xtd bit) are set by an application. They define the identifier and type of accepted received messages. If an 11-bit Identifier (Standard Frame) is used, it is programmed to $ID[28:18]$ and $ID[17:0]$ are ignored by the CAN controller. When a Data Frame with an 11-bit Identifier is received, the $ID[17:0]$ field is set to 0.

If the $RxIE$ bit is set, the $IntPnd$ bit is set when a received Data Frame is accepted and stored in the message object.

When the message handler stores a Data Frame in the message object, it stores the received Data Length Code and eight data bytes. If the Data Length Code is less than 8, the remaining bytes of the message object are overwritten by nonspecified values.

The CAN mask registers (Msk bits in **CANIFnMSK n** , $UMask$ bit in **CANIFnMCTL** register, and $MXtd$ and $MDir$ bits in **CANIFnMSK2** register) may be used ($UMask=1$) to allow groups of Data Frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications.

16.4.11 Handling of Received Message Objects

The CPU may read a received message any time via the CAN Interface registers because the data consistency is guaranteed by the message handler state machine.

Typically, the CPU first writes 0x007F to the **CAN IFn Command Mask (CANIFnCMSK)** register and then writes the number of the message object to the **CAN IFn Command Request (CANIFnCRQ)** register. That combination transfers the whole received message from the message RAM into the Message Buffer registers (**CANIFnMSK n** , **CANIFnARB n** , and **CANIFnMCTL**). Additionally, the $NewDat$ and $IntPnd$ bits are cleared in the message RAM, acknowledging that the message has been read and clearing the pending interrupt being generated by this message object.

If the message object uses masks for acceptance filtering, the arbitration bits show which of the matching messages has been received.

The actual value of $NewDat$ shows whether a new message has been received since the last time this message object was read. The actual value of $MsgLst$ shows whether more than one message has been received since the last time this message object was read. $MsgLst$ is not automatically reset.

Using a Remote Frame, the CPU may request new data from another CAN node on the CAN bus. Setting the $TxRqst$ bit of a receive object causes the transmission of a Remote Frame with the receive object's identifier. This Remote Frame triggers the other CAN node to start the transmission of the matching Data Frame. If the matching Data Frame is received before the Remote Frame could be transmitted, the $TxRqst$ bit is automatically reset. This prevents the possible loss of data when the other device on the CAN bus has already transmitted the data, slightly earlier than expected.

16.4.12 Handling of Interrupts

If several interrupts are pending, the **CAN Interrupt (CANINT)** register points to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the CPU has cleared it.

The Status Interrupt has the highest priority. Among the message interrupts, the message object's interrupt priority decreases with increasing message number. A message interrupt is cleared by clearing the message object's `IntPnd` bit. The Status Interrupt is cleared by reading the **CAN Status (CANSTS)** register.

The interrupt identifier `IntId` in the `CANINT` register indicates the cause of the interrupt. When no interrupt is pending, the register holds the value to 0. If the value of `CANINT` is different from 0, then there is an interrupt pending. If the `IE` bit is set in the `CANCTL` register, the interrupt line to the CPU is active. The interrupt line remains active until **CANINT** is 0, all interrupt sources have been cleared, (the cause of the interrupt is reset), or until `IE` is reset, which disables interrupts from the CAN controller.

The value 0x8000 in the **CANINT** register indicates that an interrupt is pending because the CAN module has updated, but not necessarily changed the **CANSTS** register (Error Interrupt or Status Interrupt). This indicates that there is either a new Error Interrupt or a new Status Interrupt. A write access can clear the `RxOK`, `TxOK`, and `LEC` flags in the **CANSTS** register, however, only a read access to the **CANSTS** register will clear the source of the status interrupt.

`IntId` points to the pending message interrupt with the highest interrupt priority. The `SIE` bit in the **CANCTL** register controls whether a change of the status register may cause an interrupt. The `EIE` bit in the **CANCTL** register controls whether any interrupt from the CAN controller actually generates an interrupt to the microcontroller's interrupt controller. The **CANINT** interrupt register is updated even when the `IE` bit is set to zero.

There are two possibilities when handling the source of a message interrupt. The first is to read the `IntId` bit in the **CANINT** interrupt register to determine the highest priority interrupt that is pending, and the second is to read the **CAN Message Interrupt Pending (CANMSGnINT)** register to see all of the message objects that have pending interrupts.

An interrupt service routine reading the message that is the source of the interrupt may read the message and reset the message object's `IntPnd` at the same time by setting the `ClrIntPnd` bit in the **CAN IFn Command Mask (CANIFnCMSK)** register. When the `IntPnd` bit is cleared, the **CANINT** register will contain the message number for the next message object with a pending interrupt.

16.4.13 Bit Timing Configuration Error Considerations

Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly. In many cases, the CAN bit synchronization amends a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. In the case of arbitration, however, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive. The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and of the CAN nodes' interaction on the CAN bus.

16.4.14 Bit Time and Bit Rate

The CAN system supports bit rates in the range of lower than 1 Kbps up to 1000 Kbps. Each member of the CAN network has its own clock generator. The timing parameter of the bit time can be configured individually for each CAN node, creating a common bit rate even though the CAN nodes' oscillator periods may be different.

Because of small variations in frequency caused by changes in temperature or voltage and by deteriorating components, these oscillators are not absolutely stable. As long as the variations

remain inside a specific oscillator's tolerance range, the CAN nodes are able to compensate for the different bit rates by periodically resynchronizing to the bit stream.

According to the CAN specification, the bit time is divided into four segments (see Figure 16-2 on page 402): the Synchronization Segment, the Propagation Time Segment, the Phase Buffer Segment 1, and the Phase Buffer Segment 2. Each segment consists of a specific, programmable number of time quanta (see Table 16-3 on page 402). The length of the time quantum (t_q), which is the basic time unit of the bit time, is defined by the CAN controller's system clock (f_{sys}) and the Baud Rate Prescaler (BRP):

$$t_q = BRP / f_{sys}$$

The CAN module's system clock f_{sys} is the frequency of its CAN module clock (CAN_CLK) input.

The Synchronization Segment $Sync_Seg$ is that part of the bit time where edges of the CAN bus level are expected to occur; the distance between an edge that occurs outside of $Sync_Seg$ and the $Sync_Seg$ is called the *phase error* of that edge.

The Propagation Time Segment $Prop_Seg$ is intended to compensate for the physical delay times within the CAN network.

The Phase Buffer Segments $Phase_Seg1$ and $Phase_Seg2$ surround the Sample Point.

The (Re-)Synchronization Jump Width (SJW) defines how far a resynchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

A given bit rate may be met by different bit-time configurations, but for the proper function of the CAN network, the physical delay times and the oscillator's tolerance range have to be considered.

Figure 16-2. CAN Bit Time

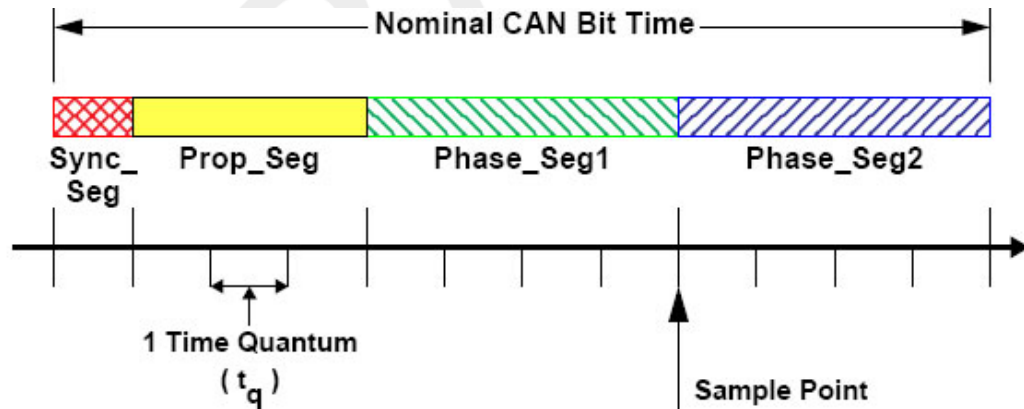


Table 16-3. CAN Protocol Ranges^a

Parameter	Range	Remark
BRP	[1 .. 32]	Defines the length of the time quantum t_q
Sync_Seg	$1 t_q$	Fixed length, synchronization of bus input to system clock
Prop_Seg	$[1 .. 8] t_q$	Compensates for the physical delay times
Phase_Seg1	$[1 .. 8] t_q$	May be lengthened temporarily by synchronization
Phase_Seg2	$[1 .. 8] t_q$	May be shortened temporarily by synchronization

Parameter	Range	Remark
SJW	[1 .. 4] t_q	May not be longer than either Phase Buffer Segment

a. This table describes the minimum programmable ranges required by the CAN protocol.

The bit timing configuration is programmed in two register bytes in the **CANBIT** register. The sum of `Prop_Seg` and `Phase_Seg1` (as `TSEG1`) is combined with `Phase_Seg2` (as `TSEG2`) in one byte, and `SJW` and `BRP` are combined in the other byte.

In these bit timing registers, the four components `TSEG1`, `TSEG2`, `SJW`, and `BRP` have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1..n], values in the range of [0..n-1] are programmed. That way, for example, `SJW` (functional range of [1..4]) is represented by only two bits. Therefore, the length of the bit time is (programmed values):

$$[TSEG1 + TSEG2 + 3] t_q$$

or (functional values):

$$[Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] t_q$$

The data in the bit timing registers are the configuration input of the CAN protocol controller. The Baud Rate Prescaler (configured by `BRP`) defines the length of the time quantum, the basic time unit of the bit time; the Bit Timing Logic (configured by `TSEG1`, `TSEG2`, and `SJW`) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the Sample Point, and occasional synchronizations are controlled by the CAN controller and are evaluated once per time quantum.

The CAN controller translates messages to and from frames. It generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. It is evaluated at the Sample Point and processes the sampled bus input bit. The time after the Sample Point that is needed to calculate the next bit to be sent (that is, the data bit, CRC bit, stuff bit, error flag, or idle) is called the Information Processing Time (IPT).

The IPT is application-specific but may not be longer than 2 t_q ; the CAN's IPT is 0 t_q . Its length is the lower limit of the programmed length of `Phase_Seg2`. In case of synchronization, `Phase_Seg2` may be shortened to a value less than IPT, which does not affect bus timing.

16.4.15 Calculating the Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a desired bit rate or bit time. The resulting bit time (1/bit rate) must be an integer multiple of the system clock period.

The bit time may consist of 4 to 25 time quanta. Several combinations may lead to the desired bit time, allowing iterations of the following steps.

The first part of the bit time to be defined is the `Prop_Seg`. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandable CAN bus systems. The resulting time for `Prop_Seg` is converted into time quanta (rounded up to the nearest integer multiple of t_q).

The `Sync_Seg` is 1 t_q long (fixed), which leaves (bit time - `Prop_Seg` - 1) t_q for the two Phase Buffer Segments. If the number of remaining t_q is even, the Phase Buffer Segments have the same length, that is, `Phase_Seg2` = `Phase_Seg1`, else `Phase_Seg2` = `Phase_Seg1` + 1.

The minimum nominal length of `Phase_Seg2` has to be regarded as well. `Phase_Seg2` may not be shorter than the CAN controller's Information Processing Time, which is, depending on the actual implementation, in the range of $[0..2] t_q$.

The length of the Synchronization Jump Width is set to its maximum value, which is the minimum of 4 and `Phase_Seg1`.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formula given below:

$$(1 - df) \times f_{nom} \leq f_{osc} \leq (1 + df) \times f_{nom}$$

where:

- `df` = maximum tolerance of oscillator frequency
- `fosc` = actual oscillator frequency
- `fnom` = nominal oscillator frequency

Maximum frequency tolerance must take into account the following formulas:

$$df \leq (\text{Phase_Seg1, Phase_Seg2})_{\min} / 2 \times (13 \times t_{\text{bit}} - \text{Phase_Seg2})$$

$$df_{\max} = 2 \times df \times f_{\text{nom}}$$

where:

- `Phase_Seg1` and `Phase_Seg2` are from Table 16-3 on page 402
- `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.

The resulting configuration is written into the **CAN Bit Timing (CANBIT)** register :

$$(\text{Phase_Seg2}-1) \& (\text{Phase_Seg1}+\text{Prop_Seg}-1) \& (\text{SynchronizationJumpWidth}-1) \& (\text{Prescaler}-1)$$

16.4.15.1 Example for Bit Timing at High Baud Rate

In this example, the frequency of `CAN_CLK` is 10 MHz, `BRP` is 0, and the bit rate is 1 Mbps.

```
tq 100 ns = tCAN_CLK
delay of bus driver 50 ns
delay of receiver circuit 30 ns
delay of bus line (40m) 220 ns
```

```

tProp 600 ns = 6 × tq
tSJW 100 ns = 1 × tq
tTSeg1 700 ns = tProp + tSJW
tTSeg2 200 ns = Information Processing Time + 1 × tq
tSync-Seg 100 ns = 1 × tq
bit time 1000 ns = tSync-Seg + tTSeg1 + tTSeg2
tolerance for CAN_CLK 0.39 % =
    min(PB1,PB2)/ 2 × (13 × bit time - PB2) =
    0.1us/ 2 × (13x 1us - 2us)

```

In the above example, the concatenated bit time parameters are (2-1)3&(7-1)4&(1-1)2&(1-1)6, and **CANBIT** is programmed to 0x1600.

16.4.15.2 Example for Bit Timing at Low Baud Rate

In this example, the frequency of **CAN_CLK** is 2 MHz, **BRP** is 1, and the bit rate is 100 Kbps.

```

tq 1 ms = 2 × tCAN_CLK
delay of bus driver 200 ns
delay of receiver circuit 80 ns
delay of bus line (40m) 220 ns
tProp 1 ms = 1 × tq
tSJW 4 ms = 4 × tq
tTSeg1 5 ms = tProp + tSJW
tTSeg2 4 ms = Information Processing Time + 3 × tq
tSync-Seg 1 ms = 1 × tq
bit time 10 ms = tSync-Seg + tTSeg1 + tTSeg2
tolerance for CAN_CLK 1.58 % =
    min(PB1,PB2)/ 2 × (13 × bit time - PB2) =
    4us/ 2 × (13 × 10us - 4us)

```

In this example, the concatenated bit time parameters are (4-1)3&(5-1)4&(4-1)2&(2-1)6, and **CANBIT** is programmed to 0x34C1.

16.5 Controller Area Network Register Map

Table 16-4 on page 405 lists the registers. All addresses given are relative to the CAN base address of:

- CAN0: 0x4004.0000

All accesses are on word (32-bit) boundaries.

Table 16-4. CAN Register Map

Offset	Name	Type	Reset	Description	See page
0x000	CANCTL	R/W	0x0000.0001	CAN Control	408
0x004	CANSTS	R/W	0x0000.0000	CAN Status	410
0x008	CANERR	RO	0x0000.0000	CAN Error Counter	413
0x00C	CANBIT	R/W	0x0000.2301	CAN Bit Timing	414
0x010	CANINT	RO	0x0000.0000	CAN Interrupt	416

Offset	Name	Type	Reset	Description	See page
0x014	CANTST	R/W	0x0000.0000	CAN Test	417
0x018	CANBRPE	R/W	0x0000.0000	CAN Baud Rate Prescaler Extension	419
0x020	CANIF1CRQ	R/W	0x0000.0001	CAN IF1 Command Request	420
0x024	CANIF1CMSK	R/W	0x0000.0000	CAN IF1 Command Mask	421
0x028	CANIF1MSK1	R/W	0x0000.FFFF	CAN IF1 Mask 1	424
0x02C	CANIF1MSK2	R/W	0x0000.FFFF	CAN IF1 Mask 2	425
0x030	CANIF1ARB1	R/W	0x0000.0000	CAN IF1 Arbitration 1	426
0x034	CANIF1ARB2	R/W	0x0000.0000	CAN IF1 Arbitration 2	427
0x038	CANIF1MCTL	R/W	0x0000.0000	CAN IF1 Message Control	428
0x03C	CANIF1DA1	R/W	0x0000.0000	CAN IF1 Data A1	430
0x040	CANIF1DA2	R/W	0x0000.0000	CAN IF1 Data A2	431
0x044	CANIF1DB1	R/W	0x0000.0000	CAN IF1 Data B1	432
0x048	CANIF1DB2	R/W	0x0000.0000	CAN IF1 Data B2	433
0x080	CANIF2CRQ	R/W	0x0000.0001	CAN IF2 Command Request	420
0x084	CANIF2CMSK	R/W	0x0000.0000	CAN IF2 Command Mask	421
0x088	CANIF2MSK1	R/W	0x0000.FFFF	CAN IF2 Mask 1	424
0x08C	CANIF2MSK2	R/W	0x0000.FFFF	CAN IF2 Mask 2	425
0x090	CANIF2ARB1	R/W	0x0000.0000	CAN IF2 Arbitration 1	426
0x094	CANIF2ARB2	R/W	0x0000.0000	CAN IF2 Arbitration 2	427
0x098	CANIF2MCTL	R/W	0x0000.0000	CAN IF2 Message Control	428
0x09C	CANIF2DA1	R/W	0x0000.0000	CAN IF2 Data A1	430
0x0A0	CANIF2DA2	R/W	0x0000.0000	CAN IF2 Data A2	431
0x0A4	CANIF2DB1	R/W	0x0000.0000	CAN IF2 Data B1	432
0x0A8	CANIF2DB2	R/W	0x0000.0000	CAN IF2 Data B2	433
0x100	CANTXRQ1	RO	0x0000.0000	CAN Transmission Request 1	434
0x104	CANTXRQ2	RO	0x0000.0000	CAN Transmission Request 2	434
0x120	CANNWDA1	RO	0x0000.0000	CAN New Data 1	435
0x124	CANNWDA2	RO	0x0000.0000	CAN New Data 2	435
0x140	CANMSG1INT	RO	0x0000.0000	CAN Message 1 Interrupt Pending	436
0x144	CANMSG2INT	RO	0x0000.0000	CAN Message 2 Interrupt Pending	436
0x160	CANMSG1VAL	RO	0x0000.0000	CAN Message 1 Valid	437
0x164	CANMSG2VAL	RO	0x0000.0000	CAN Message 2 Valid	437

16.6 Register Descriptions

The remainder of this section lists and describes the CAN registers, in numerical order by address offset. There are two sets of Interface Registers which are used to access the Message Objects in the Message RAM: **CANIF1x** and **CANIF2x**. The function of the two sets are identical and are used to queue transactions.

DRAFT

Register 1: CAN Control (CANCTL), offset 0x000

This control register initializes the module and enables test mode and interrupts.

The bus-off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or resetting `Init`. If the device goes bus-off, it sets `Init`, stopping all bus activities. Once `Init` has been cleared by the CPU, the device then waits for 129 occurrences of Bus Idle (129 * 11 consecutive High bits) before resuming normal operations. At the end of the bus-off recovery sequence, the Error Management Counters are reset.

During the waiting time after `Init` is reset, each time a sequence of 11 High bits has been monitored, a `Bit0Error` code is written to the **CANSTS** status register, enabling the CPU to readily check whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the proceeding of the bus-off recovery sequence.

CAN Control (CANCTL)

CAN0 base: 0x4004.0000
 Offset 0x000
 Type R/W, reset 0x0000.0001

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	Test	R/W	0	Test Mode Enable 0: Normal Operation 1: Test Mode
6	CCE	R/W	0	Configuration Change Enable 0: Do not allow write access to the CANBIT register. 1: Allow write access to the CANBIT register if the <code>Init</code> bit is 1.
5	DAR	R/W	0	Disable Automatic Retransmission 0: Auto retransmission of disturbed messages is enabled. 1: Auto retransmission is disabled.
4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EIE	R/W	0	Error Interrupt Enable 0: Disabled. No Error Status interrupt is generated. 1: Enabled. A change in the <code>Boff</code> or <code>EWarn</code> bits in the CANSTS register generates an interrupt.

Bit/Field	Name	Type	Reset	Description
2	SIE	R/W	0	Status Change Interrupt Enable 0: Disabled. No Status Change interrupt is generated. 1: Enabled. An interrupt is generated when a message has successfully been transmitted or received, or a CAN bus error has been detected. A change in the TxOK or RxOK bits in the CANSTS register generates an interrupt.
1	IE	R/W	0	Module Interrupt Enable 0: Interrupt disabled. 1: Interrupt enabled.
0	INIT	R/W	1	Initialization 0: Normal operation. 1: Initialization started.

DRAFT

Register 2: CAN Status (CANSTS), offset 0x004

The status register contains information for interrupt servicing such as Bus-Off, error count threshold, and error types.

The **LEC** field holds the code that indicates the type of the last error to occur on the CAN bus. This field is cleared to 0 when a message has been transferred (reception or transmission) without error. The unused error code 7 may be written by the CPU to check for updates.

An Error Interrupt is generated by the **BOff** and **EWarn** bits and a Status Change Interrupt is generated by the **RxOk**, **TxOk**, and **LEC** bits, assuming that the corresponding enable bits in the **CAN Control (CANCTL)** register are set. A change of the **EPass** bit or a write to the **RxOk**, **TxOk**, or **LEC** bits does not generate an interrupt.

Reading the **CAN Status (CANSTS)** register clears the **CAN Interrupt (CANINT)** register, if it is pending.

CAN Status (CANSTS)

CAN0 base: 0x4004.0000
Offset 0x004
Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	BOff	RO	0	Bus-Off Status 0: Module is not in bus-off state. 1: Module is in bus-off state.
6	EWarn	RO	0	Warning Status 0: Both error counters are below the error warning limit of 96. 1: At least one of the error counters has reached the error warning limit of 96.
5	EPass	RO	0	Error Passive 0: The CAN module is in the Error Active state, that is, the receive or transmit error count is less than or equal to 127. 1: The CAN module is in the Error Passive state, that is, the receive or transmit error count is greater than 127.

Bit/Field	Name	Type	Reset	Description
4	RxOK	R/W	0	<p>Received a Message Successfully</p> <p>0: Since this bit was last reset to 0, no message has been successfully received.</p> <p>1: Since this bit was last reset to 0, a message has been successfully received, independent of the result of the acceptance filtering.</p> <p>This bit is never reset by the CAN module.</p>
3	TxOK	R/W	0	<p>Transmitted a Message Successfully</p> <p>0: Since this bit was last reset to 0, no message has been successfully transmitted.</p> <p>1: Since this bit was last reset to 0, a message has been successfully transmitted error-free and acknowledged by at least one other node.</p> <p>This bit is never reset by the CAN module.</p>

DRAFT

Bit/Field	Name	Type	Reset	Description
2:0	LEC	R/W	0x0	<p>Last Error Code (type of the last error to occur on the CAN bus)</p> <p>Value Definition</p> <p>000 No Error</p> <p>001 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>010 Form Error</p> <p>A fixed format part of the received frame has the wrong format.</p> <p>011 ACK Error</p> <p>The message transmitted was not acknowledged by another node.</p> <p>100 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>101 Bit 0 Error</p> <p>A Bit 0 Error indicates that the device wanted to send a Low level (logical 0) but the monitored bus value was High (logical 1).</p> <p>During bus-off recovery, this status is set each time a sequence of 11 High bits has been monitored. This enables the CPU to monitor the proceeding of the bus-off recovery sequence without any disturbances to the bus.</p> <p>110 CRC Error</p> <p>The CRC checksum was incorrect in the received message indicate that the calculated value received did not match the calculated CRC of the data.</p> <p>111 Unused</p> <p>When the LEC bit shows this value, no CAN bus event was detected since the CPU wrote this value to LEC.</p>

Register 3: CAN Error Counter (CANERR), offset 0x008

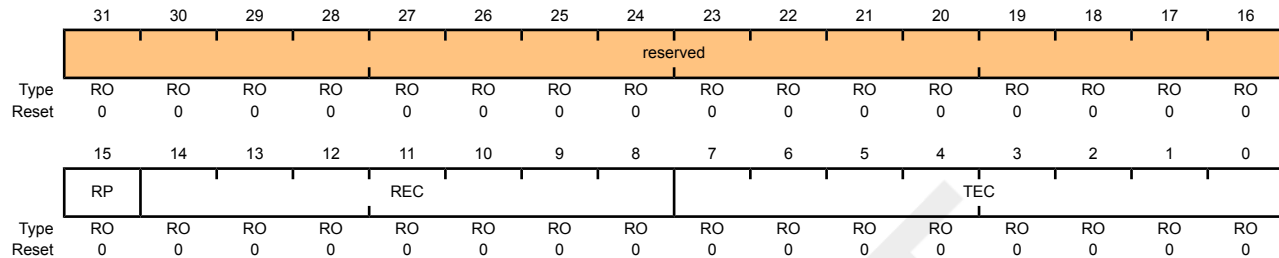
This register contains the error counter values, which can be used to analyze the cause of an error.

CAN Error Counter (CANERR)

CAN0 base: 0x4004.0000

Offset 0x008

Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	RP	RO	0	Received Error Passive 0: The Receive Error counter is below the Error Passive level (127 or less). 1: The Receive Error counter has reached the Error Passive level (128 or greater).
14:8	REC	RO	0x0	Receive Error Counter State of the receiver error counter (0 to 127).
7:0	TEC	RO	0x0	Transmit Error Counter State of the transmit error counter (0 to 255).

Register 4: CAN Bit Timing (CANBIT), offset 0x00C

This register is used to program the bit width and bit quantum. Values are to be programmed to an 8-MHz reference clock. This register is write-enabled by the `CCE` and `Init` bits in the `CANCTL` register.

With a CAN module clock (`CAN_CLK`) of 8 MHz, the register reset value of 0x230 configures the CAN for a bit rate of 500 Kbps.

CAN Bit Timing (CANBIT)

CAN0 base: 0x4004.0000
 Offset 0x00C
 Type R/W, reset 0x0000.2301

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	TSeg2			TSeg1				SJW		BRP					
Type	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14:12	TSeg2	R/W	0x2	Time Segment after Sample Point 0x00-0x07: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. So, for example, a reset value of 0x2 defines that there is 3(2+1) bit time quanta defined for <code>Phase_Seg2</code> (see Figure 16-2 on page 402). The bit time quanta is defined by BRP.
11:8	TSeg1	R/W	0x3	Time Segment Before Sample Point 0x00-0x0F: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. So, for example, the reset value of 0x3 defines that there is 4(3+1) bit time quanta defined for <code>Phase_Seg1</code> (see Figure 16-2 on page 402). The bit time quanta is define by BRP.
7:6	SJW	R/W	0x0	(Re)Synchronization Jump Width 0x00-0x03: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. During the start of frame (SOF), if the CAN controller detects a phase error (misalignment), it can adjust the length of <code>TSeg2</code> or <code>TSeg1</code> by the value in <code>SJW</code> . So the reset value of 0 adjusts the length by 1 bit time quanta.

Bit/Field	Name	Type	Reset	Description
5:0	BRP	R/W	0x1	<p>Baud Rate Prescaler</p> <p>0x00-0x03F: The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quantum. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.</p> <p>BRP defines the number of CAN clock periods that make up 1 bit time quanta, so the reset value is 2 bit time quanta (1+1).</p> <p>The BRPRE register can be used to further divide the bit time.</p>

DRAFT

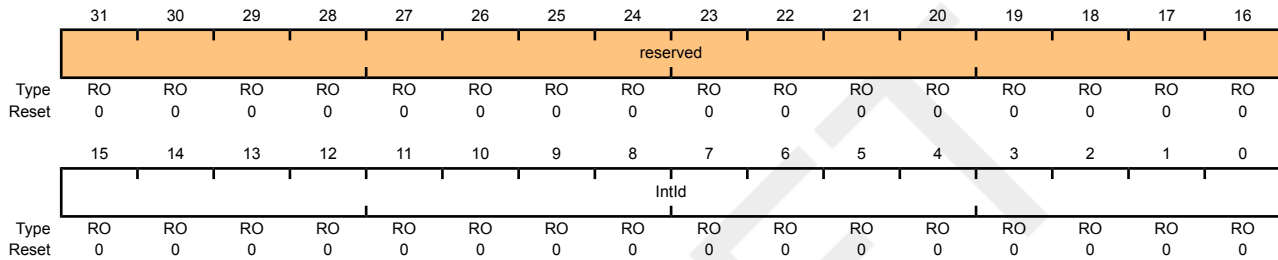
Register 5: CAN Interrupt (CANINT), offset 0x010

This register indicates the source of the interrupt.

If several interrupts are pending, the **CAN Interrupt (CANINT)** register points to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the CPU has cleared it. If the `IntId` bit is not 0x0000 (the default) and the `IE` bit in the **CANCTL** register is set, the interrupt is active. The interrupt line remains active until the `IntId` bit is set back to 0x0000 when the cause of all interrupts are reset or until `IE` is reset.

CAN Interrupt (CANINT)

CAN0 base: 0x4004.0000
 Offset 0x010
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description												
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.												
15:0	IntId	RO	0x0000	Interrupt Identifier The number in this field indicates the source of the interrupt. <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Value</th> <th>Definition</th> </tr> </thead> <tbody> <tr> <td>0x0000</td> <td>No interrupt pending</td> </tr> <tr> <td>0x0001-0x0020</td> <td>Number of the message object that caused the interrupt</td> </tr> <tr> <td>0x0021-0x7FFF</td> <td>Unused</td> </tr> <tr> <td>0x8000</td> <td>Status Interrupt</td> </tr> <tr> <td>0x8001-0xFFFF</td> <td>Unused</td> </tr> </tbody> </table>	Value	Definition	0x0000	No interrupt pending	0x0001-0x0020	Number of the message object that caused the interrupt	0x0021-0x7FFF	Unused	0x8000	Status Interrupt	0x8001-0xFFFF	Unused
Value	Definition															
0x0000	No interrupt pending															
0x0001-0x0020	Number of the message object that caused the interrupt															
0x0021-0x7FFF	Unused															
0x8000	Status Interrupt															
0x8001-0xFFFF	Unused															

Register 6: CAN Test (CANTST), offset 0x014

This is the test mode register for self-test and external pin access. It is write-enabled by the `Test` bit in the `CANCTL` register. Different test functions may be combined but when the `TX` bit is not equal to 0x0, it disturbs message transmits.

CAN Test (CANTST)

CAN0 base: 0x4004.0000

Offset 0x014

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description										
31:8	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.										
7	Rx	RO	0	Receive Observation Displays the value on the <code>CANnRx</code> pin.										
6:5	Tx	R/W	0x0	Transmit Control Overrides control of <code>CANnTx</code> pin. <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>00</td> <td><code>CAN_TX</code> is controlled by the CAN module (default)</td> </tr> <tr> <td>01</td> <td>Sample Point signal driven on the <code>CAN_TX</code> pin</td> </tr> <tr> <td>10</td> <td><code>CAN_TX</code> drives a Low value</td> </tr> <tr> <td>11</td> <td><code>CAN_TX</code> drives a High value</td> </tr> </tbody> </table>	Value	Description	00	<code>CAN_TX</code> is controlled by the CAN module (default)	01	Sample Point signal driven on the <code>CAN_TX</code> pin	10	<code>CAN_TX</code> drives a Low value	11	<code>CAN_TX</code> drives a High value
Value	Description													
00	<code>CAN_TX</code> is controlled by the CAN module (default)													
01	Sample Point signal driven on the <code>CAN_TX</code> pin													
10	<code>CAN_TX</code> drives a Low value													
11	<code>CAN_TX</code> drives a High value													
4	LBack	R/W	0	Loopback Mode 0: Disabled. 1: Enabled.										
3	Silent	R/W	0	Silent Mode Do not transmit data; monitor the bus. Also known as Bus Monitor mode. 0: Disabled. 1: Enabled.										
2	Basic	R/W	0	Basic Mode 0: Disabled. 1: Use <code>CANIF1</code> registers as transmit buffer, and use <code>CANIF2</code> registers as receive buffer.										

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

DRAFT

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 with the `CCE` bit in the `CANCTL` register.

CAN Baud Rate Prescaler Extension (CANBRPE)

CAN0 base: 0x4004.0000

Offset 0x018

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	BRPE	R/W	0x0	Baud Rate Prescaler Extension. 0x00-0x0F: Extend the <code>BRP</code> bit to values up to 1023. The actual interpretation by the hardware is one more than the value programmed by <code>BRPE</code> (MSBs) and <code>BRP</code> (LSBs) are used.

Register 8: CAN IF1 Command Request (CANIF1CRQ), offset 0x020

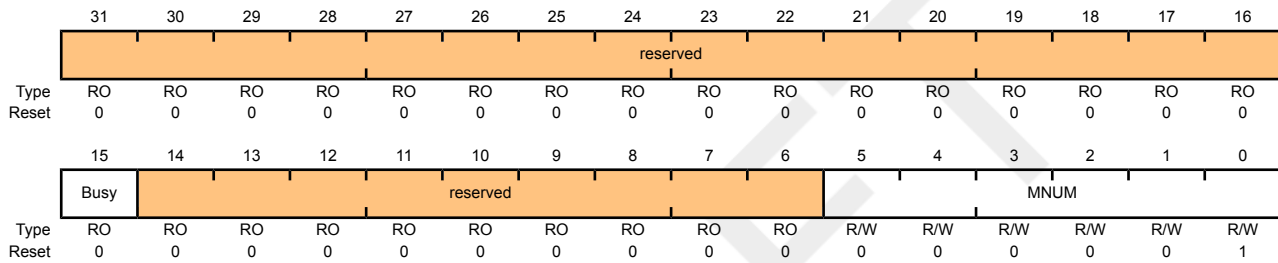
Register 9: CAN IF2 Command Request (CANIF2CRQ), offset 0x080

This register is used to start a transfer when its MNUM bit field is updated. Its Busy bit indicates that the information is transferring from the CAN Interface Registers to the internal message RAM.

A message transfer is started as soon as there is a write of the message object number with the MNUM bit. With this write operation, the Busy bit is automatically set to 1 to indicate that a transfer is in progress. After a wait time of 3 to 6 CAN_CLK periods, the transfer between the interface register and the message RAM completes, which then sets the Busy bit back to 0.

CAN IF1 Command Request (CANIF1CRQ)

CAN0 base: 0x4004.0000
 Offset 0x020
 Type 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	0x0	Busy Flag 0: Reset when read/write action has finished. 1: Set when a write occurs to the message number in this register.
14:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	MNUM	R/W	0x01	Message Number Selects one of the 32 message objects in the message RAM for data transfer. The message objects are numbered from 1 to 32.
	Value	Description		
	0x00	0 is not a valid message number; it is interpreted as 0x20, or object 32.		
	0x01-0x20	Indicates specified message object 1 to 32.		
	0x21-0x3F	Not a valid message number; values are shifted and it is interpreted as 0x01-0x1F.		

Register 10: CAN IF1 Command Mask (CANIF1CMSK), offset 0x024**Register 11: CAN IF2 Command Mask (CANIF2CMSK), offset 0x084**

The Command Mask registers specify the transfer direction and select which buffer registers are the source or target of the data transfer.

CAN IF1 Command Mask (CANIF1CMSK)

CAN0 base: 0x4004.0000

Offset 0x024

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000	Software should not rely on the value of a 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 0: Read. Transfer the message object address specified by the CAN Command Request (CANIFnCRQ) register to the CAN message buffer registers (CANIFnMSK1 , CANIFnMSK2 , CANIFnARB1 , CANIFnARB2 , CANIFnCTL , CANIFnDA1 , CANIFnDA2 , CANIFnDB1 , and CANIFnDB2). 1: Write. Transfer data from the message buffer registers to the message object address specified by the CANIFnCRQ register.
6	Mask	R/W	0x0	Access Mask Bits When WRNRD=1 (writes): 0: Mask bits unchanged. 1: Transfer IDMask + Dir + MXtd to message object. When WRNRD=0 (reads): 0: Mask bits unchanged. 1: Transfer IDMask + Dir + MXtd of the message object into the Interface Registers.
5	Arb	R/W	0x0	Access Arbitration Bits When WRNRD=1 (writes): 0: Arbitration bits unchanged. 1: Transfer ID + Dir + Xtd + MsgVal to message object. When WRNRD=0 (reads): 0: Arbitration bits unchanged. 1: Transfer ID + Dir + Xtd + MsgVal to Message Buffer Register.

Bit/Field	Name	Type	Reset	Description
4	Control	R/W	0x0	<p>Access Control Bits</p> <p>When WRNRD=1 (writes):</p> <p>0: Control bits unchanged.</p> <p>1: Transfer control bits to message object.</p> <p>When WRNRD=0 (reads):</p> <p>0: Control bits unchanged.</p> <p>1: Transfer control bits to Message Buffer Register.</p>
3	ClrIntPnd	R/W	0x0	<p>Clear Interrupt Pending Bit</p> <p>Note: This bit is not used when in write (WRNRD=1).</p> <p>0: IntPnd bit in CANIFnMCTL register remains unchanged.</p> <p>1: Clear IntPnd bit in the CANIFnMCTL register in the message object.</p>
2	TxRqst/NewDat	R/W	0x0	<p>When WRNRD=1 (writes):</p> <p>Access Transmission Request Bit</p> <p>0: TxRqst bit unchanged.</p> <p>1: Set TxRqst bit</p> <p>Note: If a transmission is requested by programming this TxRqst bit, the parallel TxRqst in the CANIFnMCTL register is ignored.</p> <p>When WRNRD=0 (reads):</p> <p>Access New Data Bit</p> <p>0: NewDat bit unchanged.</p> <p>1: Clear NewDat bit in the message object.</p> <p>Note: A read access to a message object can be combined with the reset of the control bits IntPdn and NewDat. The values of these bits that are transferred to the CANIFnMCTL register always reflect the status before resetting these bits.</p>
1	DataA	R/W	0x0	<p>Access Data Byte 0 to 3</p> <p>When WRNRD=1 (writes):</p> <p>0: Data bytes 0-3 are unchanged.</p> <p>1: Transfer data bytes 0-3 (CANIFnDA1 and CANIFnDA2) to message object.</p> <p>When WRNRD=0 (reads):</p> <p>0: Data bytes 0-3 are unchanged.</p> <p>1: Transfer data bytes 0-3 in message object to CANIFnDA1 and CANIFnDA2.</p>

Bit/Field	Name	Type	Reset	Description
0	DataB	R/W	0x0	<p>Access Data Byte 4 to 7</p> <p>When WRNRD=1 (writes):</p> <p>0: Data bytes 4-7 unchanged.</p> <p>1: Transfer data bytes 4-7 (CANIFnDB1 and CANIFnDB2) to message object.</p> <p>When WRNRD=0 (reads):</p> <p>0: Data bytes 4-7 unchanged.</p> <p>1: Transfer data bytes 4-7 in message object to CANIFnDB1 and CANIFnDB2.</p>

DRAFT

Register 12: CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028

Register 13: CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088

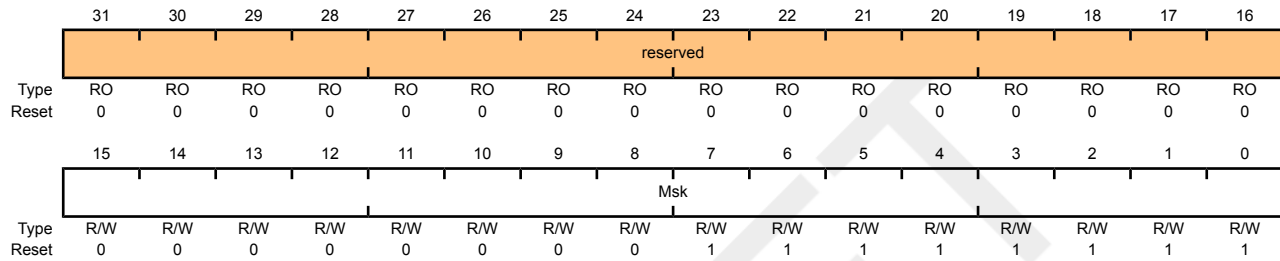
The mask information provided in this register accompanies the data (**CANIFnDAn**), arbitration information (**CANIFnARBn**), and control information (**CANIFnMCTL**) to the message object in the message RAM. The mask is used with the ID bit in the **CANIFnARBn** register for acceptance filtering. Additional mask information is contained in the **CANIFnMSK2** register.

CAN IF1 Mask 1 (CANIF1MSK1)

CAN0 base: 0x4004.0000

Offset 0x028

Type 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	0xFF	Identifier Mask 0: The corresponding identifier bit (ID) in the message object cannot inhibit the match in acceptance filtering. 1: The corresponding identifier bit (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

Offset 0x02C

Type R/W, reset 0x0000.FFFF

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	MXtd	R/W	0x1	Mask Extended Identifier 0: The extended identifier bit (<i>xtd</i> in the CANIFnARB2 register) has no effect on the acceptance filtering. 1: The extended identifier bit <i>xtd</i> is used for acceptance filtering.
14	MDir	R/W	0x1	Mask Message Direction 0: The message direction bit (<i>Dir</i> in the CANIFnARB2 register) has no effect for acceptance filtering. 1: The message direction bit <i>Dir</i> is used for acceptance filtering.
13	reserved	RO	0x1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12:0	Msk	R/W	0xFF	Identifier Mask 0: The corresponding identifier bit (ID) in the message object cannot inhibit the match in acceptance filtering. 1: The corresponding identifier bit (ID) is used for acceptance filtering.

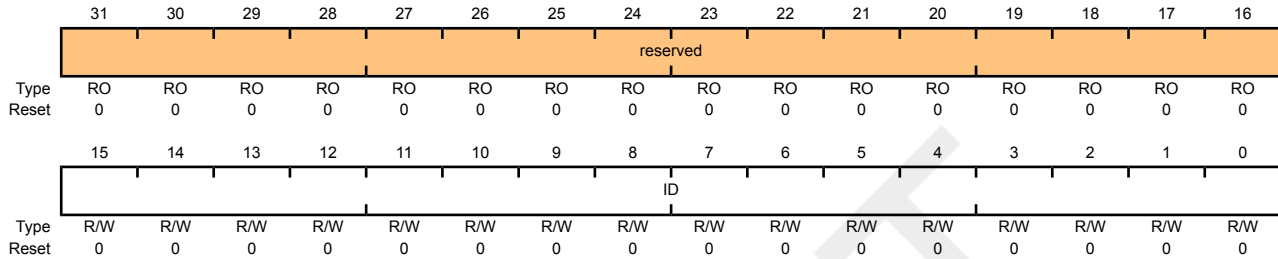
Register 16: CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030

Register 17: CAN IF2 Arbitration 1 (CANIF2ARB1), offset 0x090

This register, along with **CANIFnARB2**, holds the identifiers for acceptance filtering.

CAN IF1 Arbitration 1 (CANIF1ARB1)

CAN0 base: 0x4004.0000
 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	0x00	<p>Message Identifier</p> <p>This bit field is used with the <code>ID</code> field in the CANIFnARB2 register to create the message identifier. <code>ID[28:0]</code> is the Extended Frame and <code>ID[28:18]</code> is the Standard Frame.</p>

Register 18: CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034**Register 19: CAN IF2 Arbitration 2 (CANIF2ARB2), offset 0x094**

This register, along with **CANIFnARB1**, holds information for acceptance filtering.

CAN IF1 Arbitration 2 (CANIF1ARB2)

CAN0 base: 0x4004.0000

Offset 0x034

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
	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	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	MsgVal	R/W	0x0	<p>Message Valid</p> <p>0: The message object is ignored by the message handler.</p> <p>1: The message object is configured and will be considered by the message handler within the CAN controller.</p> <p>All unused message objects should have this bit cleared during initialization and before clearing the <code>Init</code> bit in the CANCTL register. The <code>MsgVal</code> bit must also be cleared before any of the following bits are modified or if the message object is no longer required: the <code>ID</code> bit fields in the CANIFnARBn registers, the <code>Xtd</code> and <code>Dir</code> bits in the CANIFnARB2 register, or the <code>DLC</code> bits in the CANIFnMCTL register.</p>
14	Xtd	R/W	0x0	<p>Extended Identifier</p> <p>0: The 11-bit Standard Identifier will be used for this message object.</p> <p>1: The 29-bit Extended Identifier will be used for this message object.</p>
13	Dir	R/W	0x0	<p>Message Direction</p> <p>0: Receive. On <code>TxRqst</code>, a Remote Frame with the identifier of this message object is transmitted. On reception of a Data Frame with matching identifier, that message is stored in this message object.</p> <p>1: Transmit. On <code>TxRqst</code>, the respective message object is transmitted as a Data Frame. On reception of a Remote Frame with matching identifier, <code>TxRqst</code> bit of this message object is set (if <code>RmtEn=1</code>).</p>
12:0	ID	R/W	0x0	<p>Message Identifier</p> <p>Used with the <code>ID</code> bit in the CANIFnARB1 register to create the message identifier. <code>ID[28:0]</code> is the Extended Frame and <code>ID[28:18]</code> is the Standard Frame.</p>

Register 20: CAN IF1 Message Control (CANIF1MCTL), offset 0x038

Register 21: CAN IF2 Message Control (CANIF2MCTL), offset 0x098

This register holds the control information associated with the message object to be sent to the Message RAM.

CAN IF1 Message Control (CANIF1MCTL)

CAN0 base: 0x4004.0000
 Offset 0x038
 Type 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
	NewDat	MsgLst	IntPnd	UMask	TxE	RxE	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	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	NewDat	R/W	0x0	<p>New Data</p> <p>0: No new data has been written into the data portion of this message object by the message handler since the last time this flag was cleared by the CPU.</p> <p>1: The message handler or the CPU has written new data into the data portion of this message object.</p>
14	MsgLst	R/W	0x0	<p>Message Lost</p> <p>0: No message was lost since the last time this bit was reset by the CPU.</p> <p>1: The message handler stored a new message into this object when <code>NewDat</code> was set; the CPU has lost a message.</p> <p>This bit is only valid for message objects with the <code>Dir</code> bit in the CANIFnARB2 register set to 0 (receive).</p>
13	IntPnd	R/W	0x0	<p>Interrupt Pending</p> <p>0: This message object is not the source of an interrupt.</p> <p>1: This message object is the source of an interrupt. The interrupt identifier in the CAN Interrupt (CANINT) register will point to this message object if there is not another interrupt source with a higher priority.</p>
12	UMask	R/W	0x0	<p>Use Acceptance Mask</p> <p>0: Mask ignored.</p> <p>1: Use mask (<code>Msk</code>, <code>Mxtd</code>, and <code>MDir</code>) for acceptance filtering.</p>

Bit/Field	Name	Type	Reset	Description						
11	TxIE	R/W	0x0	<p>Transmit Interrupt Enable</p> <p>0: The <code>IntPnd</code> bit in the CANIFnMCTL register is unchanged after a successful transmission of a frame.</p> <p>1: The <code>IntPnd</code> bit in the CANIFnMCTL register is set after a successful transmission of a frame.</p>						
10	RxIE	R/W	0x0	<p>Receive Interrupt Enable</p> <p>0: The <code>IntPnd</code> bit in the CANIFnMCTL register is unchanged after a successful reception of a frame.</p> <p>1: The <code>IntPnd</code> bit in the CANIFnMCTL register is set after a successful reception of a frame.</p>						
9	RmtEn	R/W	0x0	<p>Remote Enable</p> <p>0: At the reception of a Remote Frame, the <code>TxRqst</code> bit in the CANIFnMCTL register is left unchanged.</p> <p>1: At the reception of a Remote Frame, the <code>TxRqst</code> bit in the CANIFnMCTL register is set.</p>						
8	TxRqst	R/W	0x0	<p>Transmit Request</p> <p>0: This message object is not waiting for transmission.</p> <p>1: The transmission of this message object is requested and is not yet done.</p>						
7	EoB	R/W	0x0	<p>End of Buffer</p> <p>0: Message object belongs to a FIFO Buffer and is not the last message object of that FIFO Buffer.</p> <p>1: Single message object or last message object of a FIFO Buffer.</p> <p>This bit is used to concatenate two or more message objects (up to 32) to build a FIFO buffer. For a single message object (thus not belonging to a FIFO buffer), this bit must be set to 1.</p>						
6:4	reserved	RO	0x0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>						
3:0	DLC	R/W	0x0	<p>Data Length Code</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0-0x8</td> <td>Specifies the number of bytes in the Data Frame.</td> </tr> <tr> <td>0x9-0xF</td> <td>Defaults to a Data Frame with 8 bytes.</td> </tr> </tbody> </table> <p>The <code>DLC</code> bit 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 <code>DLC</code> to the value given by the received message.</p>	Value	Description	0x0-0x8	Specifies the number of bytes in the Data Frame.	0x9-0xF	Defaults to a Data Frame with 8 bytes.
Value	Description									
0x0-0x8	Specifies the number of bytes in the Data Frame.									
0x9-0xF	Defaults to a Data Frame with 8 bytes.									

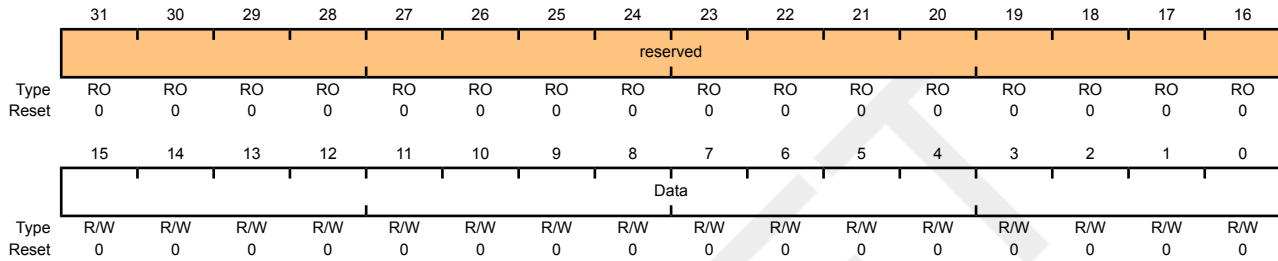
Register 22: CAN IF1 Data A1 (CANIF1DA1), offset 0x03C

Register 23: CAN IF2 Data A1 (CANIF2DA1), offset 0x09C

This register (along with **CANIFnDA2**, **CANIFnDB1**, and **CANIFnDB2**) contains the data to be sent or that has been received. In a CAN Data Frame, data byte 0 is the first byte to be transmitted or received and data byte 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte is transmitted first.

CAN IF1 Data A1 (CANIF1DA1)

CAN0 base: 0x4004.0000
 Offset 0x03C
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	Data	R/W	0x00	Data Bytes 1 and 0

Register 24: CAN IF1 Data A2 (CANIF1DA2), offset 0x040**Register 25: CAN IF2 Data A2 (CANIF2DA2), offset 0x0A0**

This register (along with **CANIFnDA1**, **CANIFnDB1**, and **CANIFnDB2**) contains 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 A2 (CANIF1DA2)

CAN0 base: 0x4004.0000

Offset 0x040

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	Data	R/W	0x00	Data Bytes 3 and 2

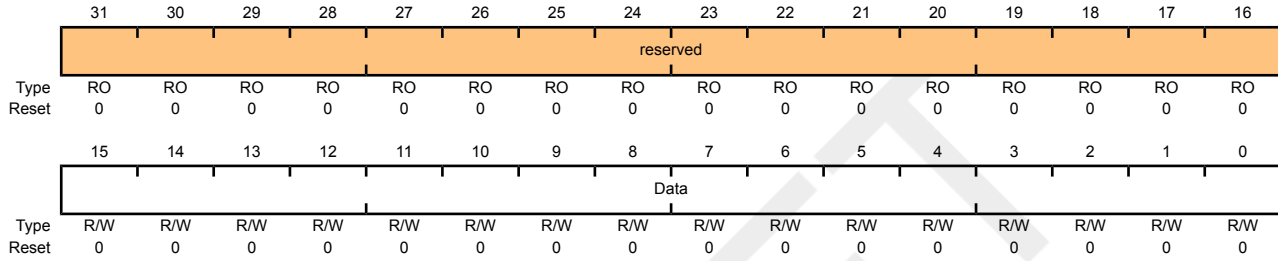
Register 26: CAN IF1 Data B1 (CANIF1DB1), offset 0x044

Register 27: CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4

This register (along with **CANIFnDA1**, **CANIFnDA2**, and **CANIFnDB2**) contains 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 B1 (CANIF1DB1)

CAN0 base: 0x4004.0000
 Offset 0x044
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	Data	R/W	0x00	Data Bytes 5 and 4

Register 28: CAN IF1 Data B2 (CANIF1DB2), offset 0x048**Register 29: CAN IF2 Data B2 (CANIF2DB2), offset 0x0A8**

This register (along with **CANIF1DA1**, **CANIF1DA2**, and **CANIF1DB1**) contains 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 B2 (CANIF1DB2)

CAN0 base: 0x4004.0000

Offset 0x048

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	Data	R/W	0x00	Data Bytes 7 and 6

Register 30: CAN Transmission Request 1 (CANTXRQ1), offset 0x100

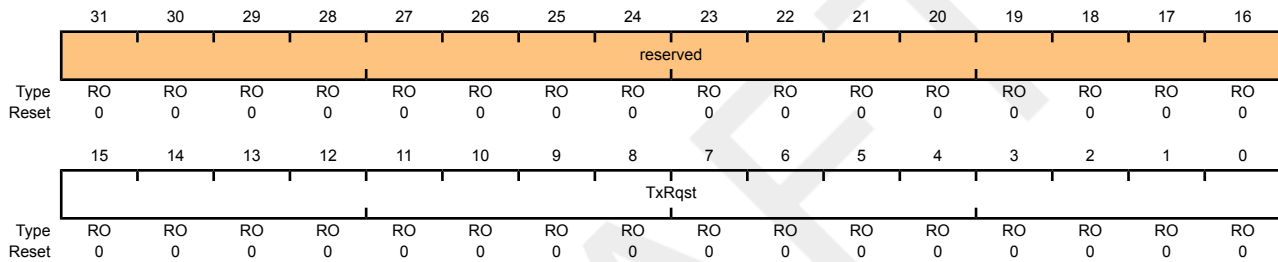
Register 31: CAN Transmission Request 2 (CANTXRQ2), offset 0x104

The **CANTXRQ1** and **CANTXRQ2** registers hold the TxRqst bits of the 32 message objects. By reading out these bits, the CPU can check which message object has a transmission request pending. The TxRqst bit of a specific message object can be changed by three sources: (1) the CPU via the **CAN IFn Message Control (CANIFnMCTL)** register, (2) the message handler state machine after the reception of a Remote Frame, or (3) the message handler state machine after a successful transmission.

The **CANTXRQ1** register contains the TxRqst bit of the first 16 message objects in the message RAM; the **CANTXRQ2** register contains the TxRqst bit of the second 16 message objects.

CAN Transmission Request 1 (CANTXRQ1)

CAN0 base: 0x4004.0000
 Offset 0x100
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TxRqst	RO	0x00	Transmission Request Bits (of all message objects) 0: The message object is not waiting for transmission. 1: The transmission of the message object is requested and is not yet done.

Register 32: CAN New Data 1 (CANNWDA1), offset 0x120**Register 33: CAN New Data 2 (CANNWDA2), offset 0x124**

The **CANNWDA1** and **CANNWDA2** registers hold the *NewDat* bits of the 32 message objects. By reading these bits, the CPU can check which message object has its data portion updated. The *NewDat* bit of a specific message object can be changed by three sources: (1) the CPU via the **CAN IFn Message Control (CANIFnMCTL)** register, (2) the message handler state machine after the reception of a Data Frame, or (3) the message handler state machine after a successful transmission.

The **CANNWDA1** register contains the *NewDat* bit of the first 16 message objects in the message RAM; the **CANNWDA2** register contains the *NewDat* bit of the second 16 message objects.

CAN New Data 1 (CANNWDA1)

CAN0 base: 0x4004.0000

Offset 0x120

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NewDat															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	NewDat	RO	0x00	New Data Bits (of all message objects) 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.

Register 34: CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140

Register 35: CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144

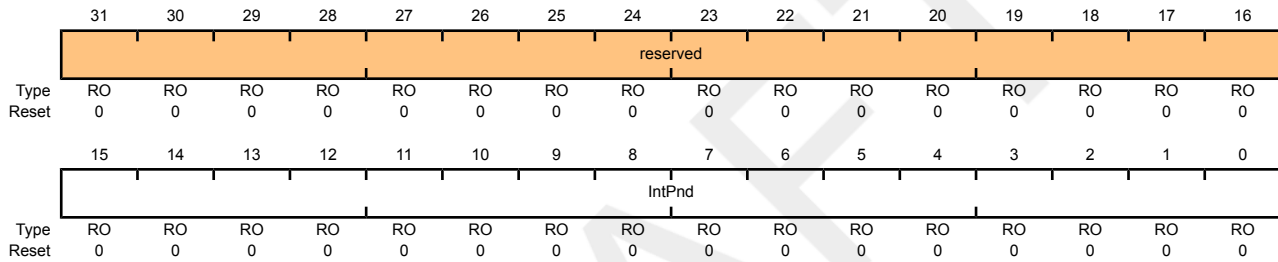
The **CANMSG1INT** and **CANMSG2INT** registers hold the `IntPnd` bits of the 32 message objects. By reading these bits, the CPU can check which message object has an interrupt pending. The `IntPnd` bit of a specific message object can be changed through two sources: (1) the CPU via the **CAN IFn Message Control (CANIFnMCTL)** register, or (2) the message handler state machine after the reception or transmission of a frame.

This field is also encoded in the **CAN Interrupt (CANINT)** register.

The **CANMSG1INT** register contains the `IntPnd` bit of the first 16 message objects in the message RAM; the **CANMSG2INT** register contains the `IntPnd` bit of the second 16 message objects.

CAN Message 1 Interrupt Pending (CANMSG1INT)

CAN0 base: 0x4004.0000
 Offset 0x140
 Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	IntPnd	RO	0x00	Interrupt Pending Bits (of all message objects) 0: This message object is not the source of an interrupt. 1: This message object is the source of an interrupt.

Register 36: CAN Message 1 Valid (CANMSG1VAL), offset 0x160**Register 37: CAN Message 2 Valid (CANMSG2VAL), offset 0x164**

The **CANMSG1VAL** and **CANMSG2VAL** registers hold the `MsgVal` bits of the 32 message objects. By reading these bits, the CPU can check which message object is valid. The message value of a specific message object can be changed with the **CAN IFn Message Control (CANIFnMCTL)** register.

The **CANMSG1VAL** register contains the `MsgVal` bit of the first 16 message objects in the message RAM; the **CANMSG2VAL** register contains the `MsgVal` bit of the second 16 message objects in the message RAM

CAN Message 1 Valid (CANMSG1VAL)

CAN0 base: 0x4004.0000

Offset 0x160

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
	MsgVal															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	MsgVal	RO	0x00	<p>Message Valid Bits (of all message objects)</p> <p>0: This message object is not configured and is ignored by the message handler.</p> <p>1: This message object is configured and should be considered by the message handler.</p>

17 Ethernet Controller

ENET

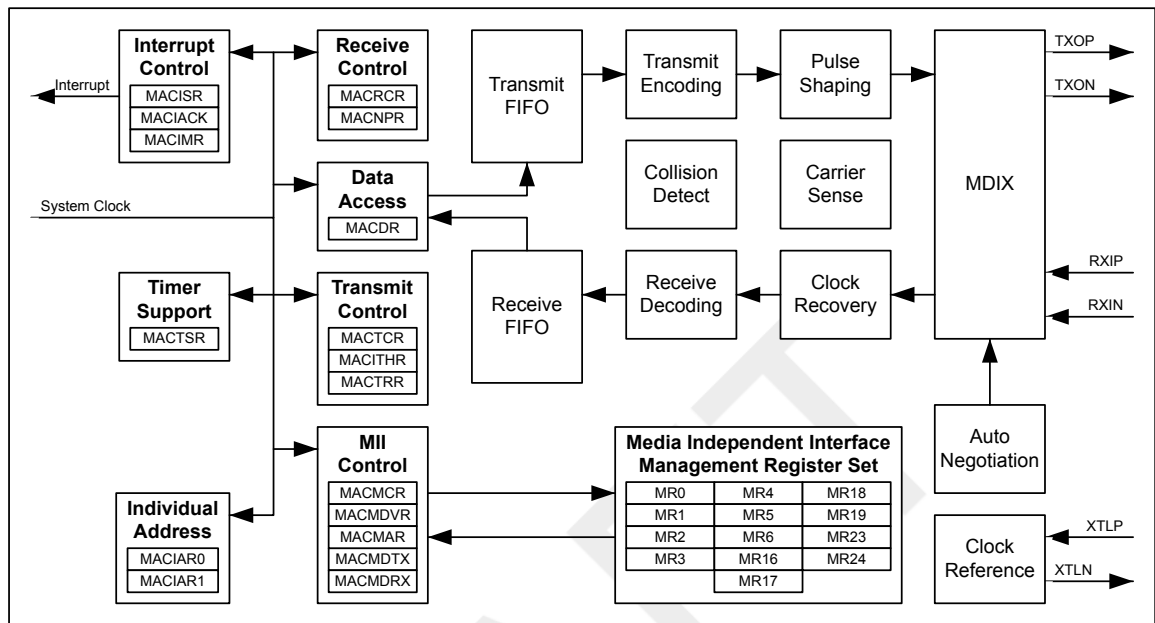
The Stellaris[®] Ethernet Controller consists of a fully integrated media access controller (MAC) and network physical (PHY) interface device. The Ethernet controller conforms to *IEEE 802.3* specifications and fully supports 10BASE-T and 100BASE-TX standards.

The Ethernet controller module has the following features:

- Conforms to the *IEEE 802.3-2002 specification*
 - 10BASE-T/100BASE-TX IEEE-802.3 compliant. Requires only a dual 1:1 isolation transformer interface to the line.
 - 10BASE-T/100BASE-TX ENDEC, 100BASE-TX scrambler/descrambler
 - Full-featured auto-negotiation
- Multiple operational modes
 - Full- and half-duplex 100 Mbps
 - Full- and half-duplex 10 Mbps
 - Power-saving and power-down modes
- Highly configurable
 - Programmable MAC address
 - LED activity selection
 - Promiscuous mode support
 - CRC error-rejection control
 - User-configurable interrupts
- Physical media manipulation
 - Automatic MDI/MDI-X cross-over correction
 - Register-programmable transmit amplitude
 - Automatic polarity correction and 10BASE-T signal reception

17.1 Block Diagram

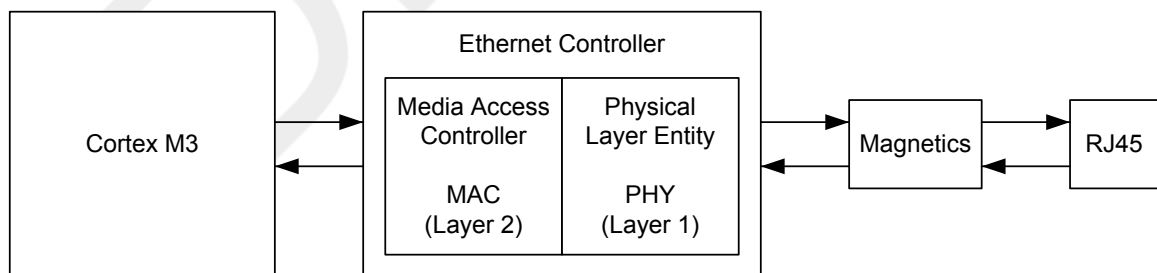
Figure 17-1. Ethernet Controller Block Diagram



17.2 Functional Description

As illustrated in Figure 17-2 on page 439, the Ethernet Controller is functionally divided into two layers or modules - the Media Access Controller (MAC) layer and Network Physical (PHY) layer. These correspond to the OSI model layers 2 and 1. The primary interface to the Ethernet controller is a simple bus interface to the MAC layer. The MAC layer provides transmit and receive processing for ethernet frames. The MAC layer also provides the interface to the PHY module via an internal Media Independent Interface (MII).

Figure 17-2. Ethernet Controller



17.2.1 Internal MII Operation

For the MII management interface to function properly, the MDIO signal must be connected through a 10k Ω pull-up resistor to the +3.3V supply. Failure to connect this pull-up resistor will prevent management transactions on this internal MII to function. Note that it is possible for data transmission across the MII to still function since the PHY layer will auto-negotiate the link parameters by default.

For the MII management interface to function properly, the internal clock must be divided down from the system clock to a frequency no greater than 2.5 MHz. The MACMDV register contains the divider used for scaling down the system clock. See page 458 for more details about the use of this register.

17.2.2 PHY Configuration/Operation

The Physical Layer (PHY) in the Ethernet controller includes integrated ENDECs, scrambler/descrambler, dual-speed clock recovery, and full-featured auto-negotiation functions. The transmitter includes an on-chip pulse shaper and a low-power line driver. The receiver has an adaptive equalizer and a baseline restoration circuit required for accurate clock and data recovery. The transceiver interfaces to Category-5 unshielded twisted pair (Cat-5 UTP) cabling for 100BASE-TX applications, and Category-3 unshielded twisted pair (Cat-3 UTP) for 10BASE-T applications. The Ethernet Controller is connected to the line media via dual 1:1 isolation transformers. No external filter is required.

17.2.2.1 Clock Selection

The PHY has an on-chip crystal oscillator which can also be driven by an external oscillator. In this mode of operation, a 25-MHz crystal should be connected between the XTLP_{PHY} and XTLN_{PHY} pins. Alternatively, an external 25-MHz clock input can be connected to the XTLP pin. In this mode of operation, a crystal is not required and the XTLN pin must be tied to ground.

17.2.2.2 Auto-Negotiation

The PHY supports the auto-negotiation functions of Clause 28 of the *IEEE 802.3* standard for 10/100 Mbps operation over copper wiring. This function can be enabled via register settings. The autonegotiation function defaults to On and bit 12 (*ANEGEN*) in the **MR0** register is High after reset. Software can disable the auto-negotiation function by writing to the *ANEGEN* bit. The contents of the **MR4** register are sent to the PHY's link partner during auto-negotiation via fast-link pulse coding.

Once auto-negotiation is complete, bits 11:10 (*DPLX* and *RATE*) in the **MR18** register reflect the actual speed and duplex that was chosen. If auto-negotiation fails to establish a link for any reason, bit 12 (*ANEGF*) in the **MR18** register reflects this and auto-negotiation restarts from the beginning. Writing a 1 to bit 9 (*RANEG*) in the **MR0** register also causes auto-negotiation to restart.

17.2.2.3 Polarity Correction

The PHY is capable of either automatic or manual polarity reversal for 10BASE-T and auto-negotiation functions. Bits 4 and 5 (*RVSPOL* and *APOL*) in the **MR16** register control this feature. The default is automatic mode, where *APOL* is Low and *RVSPOL* indicates if the detection circuitry has inverted the input signal. To enter manual mode, *APOL* should be set High and *RVSPOL* then controls the signal polarity.

17.2.2.4 MDI/MDI-X Configuration

The PHY supports the automatic MDI/MDI-X configuration as defined in IEEE 802.3 2002. This eliminates the need for cross-over cables when connecting to another device, such as a hub. The algorithm is controlled via settings in the **MR24** register. Refer to page 481 for additional details about these settings.

17.2.2.5 LED Indicators

The PHY supports two LED signals that can be used to indicate various states of operation of the Ethernet Controller. These signals are mapped to the *LED0* and *LED1* pins. By default, these pins are configured as GPIO signals (PF3 and PF2). For the PHY layer to drive these signals, they must be reconfigured to their hardware function. Refer to the GPIO chapter for additional details. The

function of these pins is programmable via the PHY layer MR23 register. Refer to page 480 for additional details on how to program these LED functions.

17.2.3 MAC Configuration/Operation

17.2.3.1 Ethernet Frame Format

Ethernet data is carried by Ethernet frames. The basic frame format is shown in Figure 17-3 on page 441.

Figure 17-3. Ethernet Frame

Preamble	SFD	Destination Address	Source Address	Length/ Type	Data	FCS
7 Bytes	1 Byte	6 Bytes	6 Bytes	2 Bytes	46 - 1500 Bytes	4 Bytes

The seven fields of the frame are transmitted from left to right. The bits within the frame are transmitted from least to most significant bit.

- **Preamble**

The Preamble field is used by the physical layer signaling circuitry to synchronize with the received frame's timing. The preamble is 7 octets long.
- **Start Frame Delimiter (SFD)**

The SFD field follows the preamble pattern and indicates the start of the frame. Its value is 1010.1011.
- **Destination Address (DA)**

This field specifies destination addresses for which the frame is intended. The LSB of the DA determines whether the address is an individual (0), or group/multicast (1) address.
- **Source Address (SA)**

The source address field identifies the station from which the frame was initiated.
- **Length/Type Field**

The meaning of this field depends on its numeric value. The first of two octets is most significant. This field can be interpreted as length or type code. The maximum length of the data field is 1500 octets. If the value of the Length/Type field is less than or equal to 1500 decimal, it indicates the number of MAC client data octets. If the value of this field is greater than or equal to 1536 decimal, then it is type interpretation. The meaning of the Length/Type field when the value is between 1500 and 1536 decimal is unspecified by the standard. The MAC module assumes type interpretation if the value of the Length/Type field is greater than 1500 decimal.
- **Data**

The data field is a sequence of 0 to 1500 octets. Full data transparency is provided so any values can appear in this field. A minimum frame size is required to properly meet the IEEE standard. If necessary, the data field is extended by appending extra bits (a pad). The pad field can have a size of 0 to 46 octets. The sum of the data and pad lengths must be a minimum of 46 octets. The MAC module automatically inserts pads if required, though it can be disabled by a register

write. For the MAC module core, data sent/received can be larger than 1500 bytes, and no Frame Too Long error is reported. Instead, a FIFO Overrun error is reported when the frame received is too large to fit into the Ethernet controller's RAM.

- Frame Check Sequence (FCS)

The frame check sequence carries the CRC (cyclic redundancy check value). The value of this field is computed over destination address, source address, length/type, data, and pad fields using the CRC-32 algorithm. The MAC module computes the FCS value one nibble at a time. For transmitted frames, this field is automatically inserted by the MAC layer, unless disabled by the CRC bit in the MACTCTL register. For received frames, this field is automatically checked. If the FCS does not pass, the frame will not be placed in the RX FIFO, unless the FCS check is disabled by the BADCRC bit in the MACRCTL register.

17.2.3.2 MAC Layer FIFOs

For Ethernet frame transmission, a 2K Byte TX FIFO is provided that can be used to store a single frame. While the IEEE 802.3 specification limits the size of an Ethernet frame's payload section to 1500 Bytes, the Ethernet controller places no such limit. The full buffer can be used, for a payload of up to 2032 bytes.

For ethernet frame reception, a 2K Byte RX FIFO is provided that can be used to store multiple frames, up to a maximum of 31 frames. If a frame is received and there is insufficient space in the RX FIFO, an overflow error will be indicated.

For details regarding the TX and RX FIFO layout, refer to Table 17-1 on page 442. Please note the following difference between TX and RX FIFO layout. For the TX FIFO, the Data Length field in the first FIFO word refers to the Ethernet frame data payload, as shown in the 5th to nth FIFO positions. For the RX FIFO, the Frame Length field is the total length of the received ethernet frame, including the FCS and Frame Length bytes. Also note that if FCS generation is disabled with the CRC bit in the MACTCTL register, the last word in the FIFO must be the FCS bytes for the frame that has been written to the FIFO.

Also note that if the length of the data payload section is not a multiple of 4, the FCS field will overlap words in the FIFO. However, for the RX FIFO, the beginning of the next frame will always be on a word boundary.

Table 17-1. TX & RX FIFO Organization

FIFO Word Read/Write Sequence	Word Bit Fields	TX FIFO (Write)	RX FIFO (Read)
1st	7:0	Data Length LSB	Frame Length LSB
	15:8	Data Length MSB	Frame Length MSB
2nd	23:16		DA oct 1
	31:24		DA oct 2
	7:0		DA oct 3
	15:8		DA oct 4
3rd	23:16		DA oct 5
	31:24		DA oct 6
	7:0		SA oct 1
	15:8		SA oct 2
4th	23:16		SA oct 3
	31:24		SA oct 4

FIFO Word Read/Write Sequence	Word Bit Fields	TX FIFO (Write)	RX FIFO (Read)
4th	7:0	SA oct 5	
	15:8	SA oct 6	
	23:16	Len/Type MSB	
	31:24	Len/Type LSB	
5th to nth	7:0	data oct n	
	15:8	data oct n+1	
	23:16	data oct n+2	
	31:24	data oct n+3	
last	7:0	FCS 1 (if CRC generation is disabled in MACTCTL)	FCS 1
	15:8	FCS 2 (if CRC generation is disabled in MACTCTL)	FCS 2
	23:16	FCS 3 (if CRC generation is disabled in MACTCTL)	FCS 3
	31:24	FCS 4 (if CRC generation is disabled in MACTCTL)	FCS 4

17.2.3.3 Ethernet Transmission Options

The ethernet controller can automatically generate and insert the Frame Check Sequence (FCS) at the end of the transmit frame. This is controlled by the CRC bit in the MACTCTL register. For test purposes, in order to generate a frame with an invalid CRC, this feature can be disabled.

The IEEE 802.3 specification requires that the ethernet frame payload section be a minimum of 46 bytes. The ethernet controller can be configured to automatically pad the data section if the payload data section loaded into the FIFO is less than the minimum 46 bytes. This feature is controlled by the PADEN bit in the MACTCTL register.

At the MAC layer, the transmitter can be configured for both full-duplex and half-duplex operation by using the DUPLEX bit in the MACTCTL register.

17.2.3.4 Ethernet Reception Options

Using the BADCRC bit in the MACRCTL register, the ethernet controller can be configured to reject incoming ethernet frames with an invalid Frame Check Sequence field.

The ethernet receiver can also be configured for Promiscuous and Multicast modes using the PRMS and AMUL fields in the MACRCTL register. If these modes are not enabled, only ethernet frames with a broadcast address, or frames matching the MAC address programmed into the MACIA0 and MACIA1 register will be placed into the RX FIFO.

17.2.4 Interrupts

The ethernet controller can generate an interrupt for one or more of the following conditions.

- A frame has been received into an empty RX FIFO.
- A frame transmission error has occurred
- A frame has been transmitted successfully.
- A frame has been received with no room in the RX FIFO (overrun).

- A frame has been received with one or more error conditions (e.g. FCS failed).
- An MII management transaction between the MAC and PHY layers has completed.
- One or more of the following PHY layer conditions occurs.
 - Auto Negotiate Complete
 - Remote Fault
 - Link Status Change
 - Link Partner Acknowledge
 - Parallel Detect Fault
 - Page Received
 - Receive Error
 - Jabber Event Detected

17.3 Initialization and Configuration

To use the Ethernet Controller, the peripheral must be enabled by setting the ETH bits in the RCGC2 register. The following steps can then be used to configure the ethernet controller for basic operation.

1. Program the MACDIV register to obtain a 2.5 MHz clock (or less) on the internal MII. Assuming a 20 MHz system clock, the MACDIV value would be 4.
2. Program the MACIA0 and MACIA1 register for address filtering.
3. Program the MACTCTL register for Auto CRC generation, padding, and full duplex operation using a value of 0x16.
4. Program the MACRCTL register to reject frames with bad FCS using a value of 0x08.
5. Enable both the Transmitter and Receive by setting the LSB in both the MACTCTL and MACRCTL register.
6. To transmit a frame, write the frame into the TX FIFO using the MACDATA register. Then set the NEWTX bit in the MACTR register to initiate the transmit process. When the NEWTX bit has been cleared, the TX FIFO will be available for the next transmit frame.
7. To receive a frame, wait for the NPR field in the MACNP register to be non-zero. Then begin reading the frame from the RX FIFO by using the MACDATA register. When the frame (including the FCS field) has been read, the NPR field should decrement by one. When there are no more frames in the RX FIFO, the NPR field will read 0.

17.4 Ethernet Register Map

Table 17-2 on page 445 lists the Ethernet MAC registers. All addresses given are relative to the Ethernet MAC base address of 0x4004.8000.

The *IEEE 802.3* standard specifies a register set for controlling and gathering status from the PHY. The registers are collectively known as the MII Management registers and are detailed in Section

22.2.4 of the *IEEE 802.3* specification. Table 17-2 on page 445 lists the MII Management registers. All addresses given are absolute and are written directly to the `REGADR` field of the `MACMCTL` register. The format of registers 0 to 15 are defined by the IEEE specification and are common to all PHY implementations. The only variance allowed is for features that may or may not be supported by a specific PHY. Registers 16 to 31 are vendor-specific registers, used to support features that are specific to a vendors PHY implementation. Vendor-specific registers not listed are reserved.

Table 17-2. Ethernet Register Map

Offset	Name	Type	Reset	Description	See page
Ethernet MAC					
0x000	MACRIS	RO	0x0000.0000	Ethernet MAC Raw Interrupt Status	447
0x000	MACIACK	W1C	0x0000.0000	Ethernet MAC Interrupt Acknowledge	449
0x004	MACIM	R/W	0x0000.007F	Ethernet MAC Interrupt Mask	450
0x008	MACRCTL	R/W	0x0000.0008	Ethernet MAC Receive Control	451
0x00C	MACTCTL	R/W	0x0000.0000	Ethernet MAC Transmit Control	452
0x010	MACDATA	R/W	0x0000.0000	Ethernet MAC Data	453
0x014	MACIA0	R/W	0x0000.0000	Ethernet MAC Individual Address 0	454
0x018	MACIA1	R/W	0x0000.0000	Ethernet MAC Individual Address 1	455
0x01C	MACTHR	R/W	0x0000.003F	Ethernet MAC Threshold	456
0x020	MACMCTL	R/W	0x0000.0000	Ethernet MAC Management Control	457
0x024	MACMDV	R/W	0x0000.0080	Ethernet MAC Management Divider	458
0x028	MACMADD	RO	0x0000.0000	Ethernet MAC Management Address	459
0x02C	MACMTXD	R/W	0x0000.0000	Ethernet MAC Management Transmit Data	460
0x030	MACMRXD	R/W	0x0000.0000	Ethernet MAC Management Receive Data	461
0x034	MACNP	RO	0x0000.0000	Ethernet MAC Number of Packets	462
0x038	MACTR	R/W	0x0000.0000	Ethernet MAC Transmission Request	463
MII Management					
0x00	MR0	R/W	0x3100	Ethernet PHY Management Register 0 – Control	464
0x01	MR1	RO	0x7849	Ethernet PHY Management Register 1 – Status	466
0x02	MR2	RO	0x000E	Ethernet PHY Management Register 2 – PHY Identifier 1	468
0x03	MR3	RO	0x7237	Ethernet PHY Management Register 3 – PHY Identifier 2	469
0x04	MR4	R/W	0x01E1	Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement	470
0x05	MR5	RO	0x0000	Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability	472
0x06	MR6	RO	0x0000	Ethernet PHY Management Register 6 – Auto-Negotiation Expansion	473

Offset	Name	Type	Reset	Description	See page
0x10	MR16	R/W	0x0140	Ethernet PHY Management Register 16 – Vendor-Specific	474
0x11	MR17	R/W	0x0000	Ethernet PHY Management Register 17 – Interrupt Control/Status	476
0x12	MR18	RO	0x0000	Ethernet PHY Management Register 18 – Diagnostic	478
0x13	MR19	R/W	0x4000	Ethernet PHY Management Register 19 – Transceiver Control	479
0x17	MR23	R/W	0x0010	Ethernet PHY Management Register 23 – LED Configuration	480
0x18	MR24	R/W	0x00C0	Ethernet PHY Management Register 24 –MDI/MDIX Control	481

17.5 Ethernet MAC Register Descriptions

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

Register 1: Ethernet MAC Raw Interrupt Status (MACRIS), offset 0x000

The **MACRIS** register is the interrupt status register. On a read, this register gives the current status value of the corresponding interrupt prior to masking.

Ethernet MAC Raw Interrupt Status (MACRIS)

Base 0x4004.8000

Offset 0x000

Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PHYINT	RO	0x0	When set, indicates that an enabled interrupt in the PHY layer has occurred. MR17 in the PHY must be read to determine the specific PHY event that triggered this interrupt.
5	MDINT	RO	0x0	When set, indicates that a transaction (read or write) on the MII interface has completed successfully.
4	RXER	RO	0x0	This bit indicates that an error was encountered on the receiver. The possible errors that can cause this interrupt bit to be set are: <ul style="list-style-type: none"> ■ A receive error occurs during the reception of a frame (100 Mb/s 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.
3	FOV	RO	0x0	When set, indicates that an overrun was encountered on the receive FIFO.
2	TXEMP	RO	0x0	When set, indicates that the packet was transmitted and that the TX FIFO is empty.
1	TXER	RO	0x0	When set, indicates that an error was encountered on the transmitter. The possible errors that can cause this interrupt bit to be set are: <ul style="list-style-type: none"> ■ The data length field stored in the TX FIFO exceeds 2032. The frame is not sent when this error occurs. ■ The retransmission attempts during the backoff process have exceeded the maximum limit of 16.

Bit/Field	Name	Type	Reset	Description
0	RXINT	RO	0x0	When set, indicates that at least one packet has been received and is stored in the receiver FIFO.

DRAFT

Register 2: Ethernet MAC Interrupt Acknowledge (MACIACK), offset 0x000

A write of a 1 to any bit position of this register clears the corresponding interrupt bit in the **Ethernet MAC Raw Interrupt Status (MACRIS)** register.

Ethernet MAC Interrupt Acknowledge (MACIACK)

Base 0x4004.8000

Offset 0x000

Type W1C, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:6	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.
6	PHYINT	W1C	0x0	A write of a 1 to the <code>PHYINT</code> bit clears the PHYINT interrupt read from the MACRIS register.
5	MDINT	W1C	0x0	A write of a 1 to the <code>MDINT</code> bit clears the MDINT interrupt read from the MACRIS register.
4	RXER	W1C	0x0	A write of a 1 to the <code>RXER</code> bit clears the RXER interrupt read from the MACRIS register.
3	FOV	W1C	0x0	A write of a 1 to the <code>FOV</code> bit clears the FOV interrupt read from the MACRIS register.
2	TXEMP	W1C	0x0	A write of a 1 to the <code>TXEMP</code> bit clears the TXEMP interrupt read from the MACRIS register.
1	TXER	W1C	0x0	A write of a 1 to the <code>TXER</code> bit clears the TXER interrupt read from the MACRIS register and resets the TX FIFO write pointer.
0	RXINT	W1C	0x0	A write of a 1 to the <code>RXINT</code> bit clears the RXINT interrupt read from the MACRIS register.

Register 3: Ethernet MAC Interrupt Mask (MACIM), offset 0x004

This register allows software to enable/disable Ethernet MAC interrupts. Writing a 0 disables the interrupt, while writing a 1 enables it.

Ethernet MAC Interrupt Mask (MACIM)

Base 0x4004.8000

Offset 0x004

Type R/W, reset 0x0000.007F

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

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PHYINTM	R/W	1	The PHYINTM bit masks the PHYINT bit in the MACRIS register from being asserted.
5	MDINTM	R/W	1	The MDINTM bit masks the MDINT bit in the MACRIS register from being asserted.
4	RXERM	R/W	1	The RXERM bit masks the RXER bit in the MACRIS register from being asserted.
3	FOVM	R/W	1	The FOVM bit masks the FOV bit in the MACRIS register from being asserted.
2	TXEMPM	R/W	1	The TXEMPM bit masks the TXEMP bit in the MACRIS register from being asserted.
1	TXERM	R/W	1	The TXERM bit masks the TXER bit in the MACRIS register from being asserted.
0	RXINTM	R/W	1	The RXINTM bit masks the RXINT bit in the MACRIS register from being asserted.

Register 4: Ethernet MAC Receive Control (MACRCTL), offset 0x008

This register enables software to configure the receive module and control the types of frames that are received from the physical medium. It is important to note that when the receive module is enabled, all valid frames with a broadcast address of FF-FF-FF-FF-FF-FF in the Destination Address field will be received and stored in the RX FIFO, even if the `AMUL` bit is not set.

Ethernet MAC Receive Control (MACRCTL)

Base 0x4004.8000
Offset 0x008
Type R/W, reset 0x0000.0008

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

Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	RSTFIFO	R/W	0x0	When set, clears the receive FIFO. This should be done when software initialization is performed. It is recommended that the receiver be disabled (<code>RXEN = 0</code>), and then the reset initiated (<code>RSTFIFO = 1</code>). This sequence will flush and reset the RX FIFO.
3	BADCRC	R/W	0x1	The <code>BADCRC</code> bit enables the rejection of frames with an incorrectly calculated CRC.
2	PRMS	R/W	0x0	The <code>PRMS</code> bit enables Promiscuous mode, which accepts all valid frames, regardless of the Destination Address.
1	AMUL	R/W	0x0	The <code>AMUL</code> bit enables the reception of multicast frames from the physical medium.
0	RXEN	R/W	0x0	The <code>RXEN</code> bit enables the Ethernet receiver. When this bit is Low, the receiver is disabled and all frames on the physical medium are ignored.

Register 5: Ethernet MAC Transmit Control (MACTCTL), offset 0x00C

This register enables software to configure the transmit module, and control frames are placed onto the physical medium.

Ethernet MAC Transmit Control (MACTCTL)

Base 0x4004.8000

Offset 0x00C

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	DUPLEX	R/W	0x0	When set, enables Duplex mode, allowing simultaneous transmission and reception.
3	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	CRC	R/W	0x0	When set, enables the automatic generation of the CRC and the placement at the end of the packet. If this bit is not set, the frames placed in the TX FIFO will be sent exactly as they are written into the FIFO.
1	PADEN	R/W	0x0	When set, enables the automatic padding of packets that do not meet the minimum frame size.
0	TXEN	R/W	0x0	When set, enables the transmitter. When this bit is 0, the transmitter is disabled.

Register 6: Ethernet MAC Data (MACDATA), offset 0x010

This register enables software to access the TX and RX FIFOs.

Reads from this register return the data stored in the RX FIFO from the location indicated by the read pointer.

Writes to this register store the data in the TX FIFO at the location indicated by the write pointer. The write pointer is then auto-incremented to the next TX FIFO location.

There is no mechanism for randomly accessing bytes in either the RX or TX FIFOs. Data must be read from the RX FIFO sequentially and stored in a buffer for further processing. Once a read has been performed, the data in the FIFO cannot be re-read. Data must be written to the TX FIFO sequentially. If an error is made in placing the frame into the TX FIFO, the write pointer can be reset to the start of the TX FIFO by writing the `TXER` bit of the **MACIACK** register and the data re-written.

Ethernet MAC Data (MACDATA)

Base 0x4004.8000
Offset 0x010
Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	RXDATA	R	0x0	The <code>RXDATA</code> bits represent the next four bytes of data stored in the RX FIFO.

Ethernet MAC Data (MACDATA)

Base 0x4004.8000
Offset 0x010
Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	TXDATA	W	0x0	The <code>TXDATA</code> bits represent the next four bytes of data to place in the TX FIFO for transmission.

Register 7: Ethernet MAC Individual Address 0 (MACIA0), offset 0x014

This register enables software to program the first four bytes of the hardware MAC Address of the Network Interface Card (NIC). The 6-byte IAR is compared against the incoming Destination Address fields to determine whether the frame should be received.

Ethernet MAC Individual Address 0 (MACIA0)

Base 0x4004.8000

Offset 0x014

Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	MACOCT4								MACOCT3							
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
	MACOCT2								MACOCT1							
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	MACOCT4	R/W	0x0	The MACOCT4 bits represent the fourth octet of the MAC address used to uniquely identify each Ethernet Controller.
23:16	MACOCT3	R/W	0x0	The MACOCT3 bits represent the third octet of the MAC address used to uniquely identify each Ethernet Controller.
15:8	MACOCT2	R/W	0x0	The MACOCT2 bits represent the second octet of the MAC address used to uniquely identify each Ethernet Controller.
7:0	MACOCT1	R/W	0x0	The MACOCT1 bits represent the first octet of the MAC address used to uniquely identify each Ethernet Controller.

Register 8: Ethernet MAC Individual Address 1 (MACIA1), offset 0x018

This register enables software to program the last two bytes of the hardware MAC Address of the Network Interface Card (NIC). The 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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MACOCT6								MACOCT5							
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	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	MACOCT6	R/W	0x0	The MACOCT6 bits represent the sixth octet of the MAC address used to uniquely identify each Ethernet Controller.
7:0	MACOCT5	R/W	0x0	The MACOCT5 bits represent the fifth octet of the MAC address used to uniquely identify each Ethernet Controller.

Register 9: Ethernet MAC Threshold (MACTHR), offset 0x01C

This register enables software to set the threshold level at which the transmission of the frame begins. If the THRESH bits are set to 0x3F, which is the reset value, transmission does not start until the NEWTX bit is set in the MACTR register. This effectively disables the early transmission feature.

Writing the THRESH bits to any value besides all 1s enables the early transmission feature. Once the byte count of data in the TX FIFO reaches this level, transmission of the frame begins. When THRESH is set to all 0s, transmission of the frame begins after 4 bytes (a single write) are stored in the TX FIFO. Each increment of the THRESH bit field waits for an additional 32 bytes of data (eight writes) to be stored in the TX FIFO. Therefore, a value of 0x01 would wait for 36 bytes of data to be written while a value of 0x02 would wait for 68 bytes to be written. In general, early transmission starts when:

$$\text{Number of Bytes} \geq 4 (\text{THRESH} \times 8 + 1)$$

Reaching the threshold level has the same effect as setting the NEWTX bit in the MACTR register. Transmission of the frame begins and then the number of bytes indicated by the Data Length field is sent out on the physical medium. Because under-run checking is not performed, it is possible that the tail pointer may reach and pass the write pointer in the TX FIFO. This causes indeterminate values to be written to the physical medium rather than the end of the frame. Therefore, sufficient bus bandwidth for writing to the TX FIFO must be guaranteed by the software.

If a frame smaller than the threshold level needs to be sent, the NEWTX bit in the MACTR register must be set with an explicit write. This initiates the transmission of the frame even though the threshold limit has not been reached.

If the threshold level is set too small, it is possible for the transmitter to underrun. If this occurs, the transmit frame will be aborted, and a transmit error will be indicated.

Ethernet MAC Threshold (MACTHR)

Base 0x4004.8000
 Offset 0x01C
 Type R/W, reset 0x0000.003F

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

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	THRESH	R/W	0x3F	The THRESH bits represent the early transmit threshold. Once the amount of data in the TX FIFO exceeds this value, transmission of the packet begins.

Register 10: Ethernet MAC Management Control (MACMCTL), offset 0x020

This register enables software to control the transfer of data to and from the MII Management Registers in the Ethernet PHY. The address, name, type, reset configuration, and functional description of each of these registers can be found in Table 17-2 on page 445 and “MII Management Register Descriptions” on page 463.

In order to initiate a *read* transaction from the MII Management registers, the `WRITE` bit must be written with a 0 during the same cycle that the `START` bit is written with a 1.

In order to initiate a *write* transaction to the MII Management registers, the `WRITE` bit must be written with a 1 during the same cycle that the `START` bit is written with a 1.

Ethernet MAC Management Control (MACMCTL)

Base 0x4004.8000
Offset 0x020
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								REGADR				reserved	WRITE	START	
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	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
31:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:3	REGADR	R/W	0x0	The <code>REGADR</code> bit field represents the MII Management register address for the next MII management interface transaction.
2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	WRITE	R/W	0x0	The <code>WRITE</code> bit represents the operation of the next MII management interface transaction. If <code>WRITE</code> is set, the next operation will be a write; otherwise, it will be a read.
0	START	R/W	0x0	The <code>START</code> bit represents the initiation of the next MII management interface transaction. When a 1 is written to this bit, the MII register located at <code>REGADR</code> will be read (<code>WRITE=0</code>) or written (<code>WRITE=1</code>).

Register 11: Ethernet MAC Management Divider (MACMDV), offset 0x024

This register enables software to set the clock divider for the Management Data Clock (MDC). This clock is used to synchronize read and write transactions between the system and the MII Management registers. The frequency of the MDC clock can be calculated from the following formula:

$$F_{\text{mdc}} = F_{\text{ipclk}} / (2 * (\text{MACMDVR} + 1))$$

The clock divider must be written with a value that ensures that the MDC clock will not exceed a frequency of 2.5 MHz.

Ethernet MAC Management Divider (MACMDV)

Base 0x4004.8000
 Offset 0x024
 Type R/W, reset 0x0000.0080

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIV	R/W	0x80	The DIV bits are used to set the clock divider for the MDC clock used to transmit data between the MAC and PHY over the serial MII interface.

Register 12: Ethernet MAC Management Address (MACMADD), offset 0x028

This register enables software to choose the address of the PHY for the next MII Management register transaction.

Ethernet MAC Management Address (MACMADD)

Base 0x4004.8000

Offset 0x028

Type RO, reset 0x0000.0000

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

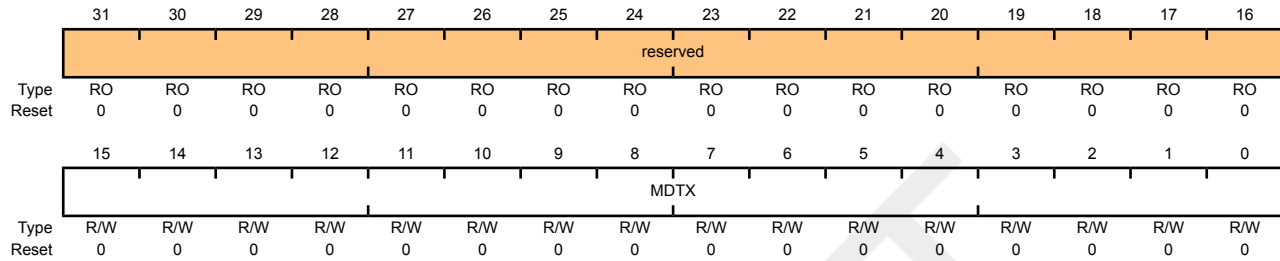
Bit/Field	Name	Type	Reset	Description
31:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 13: Ethernet MAC Management Transmit Data (MACMTXD), offset 0x02C

This register holds the next value to be written to the MII Management registers.

Ethernet MAC Management Transmit Data (MACMTXD)

Base 0x4004.8000
 Offset 0x02C
 Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MDTX	R/W	0x0	The MDTX bits represent the data that will be written in the next MII management transaction.

Register 14: 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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MDRX															
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	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MDRX	R/W	0x0	The MDRX bits represent the data that was read in the previous MII management transaction.

Register 15: 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 all 0s, there are no frames in the RX FIFO and the *RXINT* bit is not set. When *NPR* is any other value, there is at least one frame in the RX FIFO and the *RXINT* bit is set.

Ethernet MAC Number of Packets (MACNP)

Base 0x4004.8000
 Offset 0x034
 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											NPR				
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0x0	Software should not rely on the value of 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	0x0	The <i>NPR</i> bits represent the number of packets stored in the RX FIFO. While <i>NPR</i> is greater than 0, the <i>RXINT</i> interrupt will be asserted.

Register 16: Ethernet MAC Transmission Request (MACTR), offset 0x038

This register enables software to initiate the transmission of the frame currently located in the TX FIFO to the physical medium. Once the frame has been transmitted to the medium from the TX FIFO or a transmission error has been encountered, the `NEWTX` bit is auto-cleared by the hardware.

Ethernet MAC Transmission Request (MACTR)

Base 0x4004.8000
Offset 0x038
Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	NEWTX	R/W	0x0	When set, the <code>NEWTX</code> bit initiates an Ethernet transmission once the packet has been placed in the TX FIFO. This bit is cleared once the transmission has been completed. If early transmission is being used (see the <code>MACTHR</code> register), this bit does not need to be set.

17.6 MII Management Register Descriptions

The *IEEE 802.3 standard* specifies a register set for controlling and gathering status from the PHY. The registers are collectively known as the MII Management registers. All addresses given are absolute. Addresses not listed are reserved.

Register 17: Ethernet PHY Management Register 0 – Control (MR0), offset 0x00

This register enables software to configure the operation of the PHY. The default settings of these registers are designed to initialize the PHY to a normal operational mode without configuration.

Ethernet PHY Management Register 0 – Control (MR0)

Base 0x4004.8000

Offset 0x00

Type R/W, reset 0x3100

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RESET	LOOPBK	SPEEDSL	ANEGEN	PWRDN	ISO	RANEG	DUPLEX	COLT	reserved						
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	RESET	R/W	0	Reset Registers When set, resets the registers to their default state and reinitializes internal state machines. Once the reset operation has completed, this bit is cleared by hardware.
14	LOOPBK	R/W	0	Loopback Mode When set, enables the Loopback mode of operation. The receive circuitry is isolated from the physical medium and transmissions are sent back through the receive circuitry instead of the medium.
13	SPEEDSL	R/W	1	Speed Select 1: Enables the 100 Mb/s mode of operation (100BASE-TX). 0: Enables the 10 Mb/s mode of operation (10BASE-T).
12	ANEGEN	R/W	1	Auto-Negotiation Enable When set, enables the Auto-Negotiation process.
11	PWRDN	R/W	0	Power Down When set, places the PHY into a low-power consuming state.
10	ISO	R/W	0	Isolate When set, isolates transmit and receive data paths and ignores all signaling on these buses.
9	RANEG	R/W	0	Restart Auto-Negotiation When set, restarts the Auto-Negotiation process. Once the restart has initiated, this bit is cleared by hardware.

Bit/Field	Name	Type	Reset	Description
8	DUPLEX	R/W	1	Set Duplex Mode 1: Enables the Full-Duplex mode of operation. This bit can be set by software in a manual configuration process or by the Auto-Negotiation process. 0: Enables the Half-Duplex mode of operation.
7	COLT	R/W	0	Collision Test When set, enables the Collision Test mode of operation. The COLT bit asserts after the initiation of a transmission and de-asserts once the transmission is halted.
6:0	reserved	R/W	0x00	Write as 0, ignore on read.

DRAFT

Register 18: Ethernet PHY Management Register 1 – Status (MR1), offset 0x01

This register enables software to determine the capabilities of the PHY and perform its initialization and operation appropriately.

Ethernet PHY Management Register 1 – Status (MR1)

Base 0x4004.8000

Offset 0x01

Type RO, reset 0x7849

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	100X_F	100X_H	10T_F	10T_H	reserved				MFPS	ANEGC	RFAULT	ANEGA	LINK	JAB	EXTD
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RC	RO	RO	RC	RO	
Reset	0	1	1	1	1	0	0	0	0	1	0	0	1	0	0	1

Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	100X_F	RO	1	100BASE-TX Full-Duplex Mode When set, indicates that the PHY is capable of supporting 100BASE-TX Full Duplex mode.
13	100X_H	RO	1	100BASE-TX Half-Duplex Mode When set, indicates that the PHY is capable of supporting 100BASE-TX Half-Duplex mode.
12	10T_F	RO	1	10BASE-T Full-Duplex Mode When set, indicates that the PHY is capable of 10BASE-T Full-Duplex mode.
11	10T_H	RO	1	10BASE-T Half-Duplex Mode When set, indicates that the PHY is capable of supporting 10BASE-T Half-Duplex mode.
10:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	MFPS	RO	1	Management Frames with Preamble Suppressed When set, indicates that the Management Interface is capable of receiving management frames with the preamble suppressed.
5	ANEGC	RO	0	Auto-Negotiation Complete When set, indicates that the Auto-Negotiation process has been completed and that the extended registers defined by the Auto-Negotiation protocol are valid.

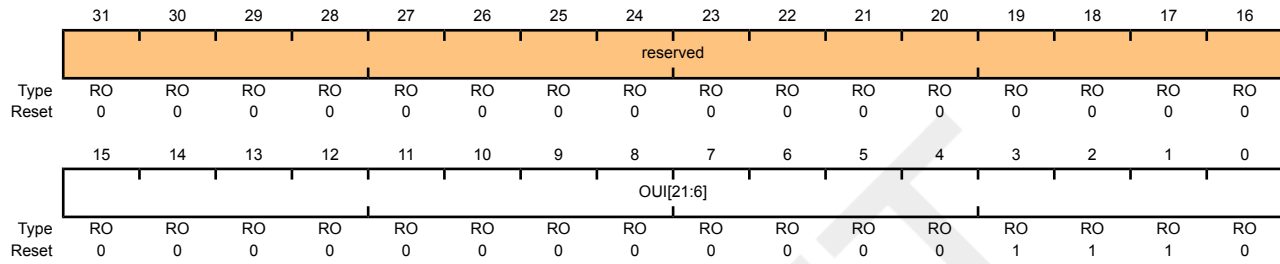
Bit/Field	Name	Type	Reset	Description
4	RFAULT	RC	0	Remote Fault When set, indicates that a remote fault condition has been detected. This bit remains set until it is read, even if the condition no longer exists.
3	ANEGA	RO	1	Auto-Negotiation When set, indicates that the PHY has the ability to perform Auto-Negotiation.
2	LINK	RO	0	Link Made When set, indicates that a valid link has been established by the PHY.
1	JAB	RC	0	Jabber Condition When set, indicates that a jabber condition has been detected by the PHY. This bit remains set until it is read, even if the jabber condition no longer exists.
0	EXTD	RO	1	Extended Capabilities When set, indicates that the PHY provides an extended set of capabilities that can be accessed through the extended register set.

Register 19: Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2), offset 0x02

This register, along with Management Register 3, provides a 32-bit value indicating the manufacturer, model, and revision information.

Ethernet PHY Management Register 2 – PHY Identifier 1 (MR2)

Base 0x4004.8000
 Offset 0x02
 Type RO, reset 0x000E



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	OUI[21:6]	RO	0x000E	Organizationally Unique Identifier[21:6] This field, along with the OUI[5:0] field in Management Register 3 , makes up the Organizationally Unique Identifier indicating the PHY manufacturer.

Register 20: Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3), offset 0x03

This register, along with Management Register 2, provides a 32-bit value indicating the manufacturer, model, and revision information.

Ethernet PHY Management Register 3 – PHY Identifier 2 (MR3)

Base 0x4004.8000

Offset 0x03

Type RO, reset 0x7237

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:10	OUI[5:0]	RO	0x1C	Organizationally Unique Identifier[5:0] This field, along with the OUI [21 : 6] field in Management Register 2 , makes up the Organizationally Unique Identifier indicating the PHY manufacturer.
9:4	MN	RO	0x23	Model Number The MN field represents the Model Number of the PHY.
3:0	RN	RO	0x7	Revision Number The RN field represents the Revision Number of the PHY.

Register 21: Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4), offset 0x04

This register provides the advertised abilities of the PHY used during Auto-Negotiation. Bits 12:5 represent the Technology Ability Field bits, A[7:0]. This field can be overwritten by software to Auto-Negotiate to an alternate common technology. Writing to this register has no effect until Auto-Negotiation is re-initiated.

Ethernet PHY Management Register 4 – Auto-Negotiation Advertisement (MR4)

Base 0x4004.8000

Offset 0x04

Type R/W, reset 0x01E1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	NP	reserved	RF	reserved				A3	A2	A1	A0	S[4:0]				
Type	RO	RO	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	NP	RO	0	Next Page When set, indicates the PHY is capable of Next Page exchanges to provide more detailed information on the PHY's capabilities.
14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	RF	R/W	0	Remote Fault When set, indicates to the link partner that a Remote Fault condition has been encountered.
12:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	A3	R/W	1	Technology Ability Field[3] When set, indicates that the PHY supports the 100Base-TX Full-Duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be written to 0 and Auto-Negotiation re-initiated with the RANEG bit.
7	A2	R/W	1	Technology Ability Field[2] When set, indicates that the PHY supports the 100Base-T half-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be written to 0 and Auto-Negotiation re-initiated.

Bit/Field	Name	Type	Reset	Description
6	A1	R/W	1	Technology Ability Field[1] When set, indicates that the PHY supports the 10Base-T Full-Duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be written to 0 and Auto-Negotiation re-initiated.
5	A0	R/W	1	Technology Ability Field[0] When set, indicates that the PHY supports the 10Base-T half-duplex signaling protocol. If software wants to ensure that this mode is not used, this bit can be written to 0 and Auto-Negotiation re-initiated.
4:0	S[4:0]	RO	0x01	Selector Field The S[4:0] field encodes 32 possible messages for communicating between PHYs. This field is hard-coded to 0x01, indicating that the Stellaris [®] PHY is IEEE 802.3 compliant.

Register 22: Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5), offset 0x05

This register provides the advertised abilities of the link partner's PHY that are received and stored during Auto-Negotiation.

Ethernet PHY Management Register 5 – Auto-Negotiation Link Partner Base Page Ability (MR5)

Base 0x4004.8000

Offset 0x05

Type RO, reset 0x0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
	reserved																
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	NP	ACK	RF	A[7:0]								S[4:0]					
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														
31:16	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	NP	RO	0	Next Page When set, indicates that the link partner's PHY is capable of Next page exchanges to provide more detailed information on the PHY's capabilities.														
14	ACK	RO	0	Acknowledge When set, indicates that the device has successfully received the link partner's advertised abilities during Auto-Negotiation.														
13	RF	RO	0	Remote Fault Used as a standard transport mechanism for transmitting simple fault information.														
12:5	A[7:0]	RO	0x00	Technology Ability Field The <i>A</i> field encodes individual technologies that are supported by the PHY. See the MR4 register.														
4:0	S[4:0]	RO	0x00	Selector Field The <i>S</i> field encodes possible messages for communicating between PHYs. <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>0x00</td> <td>Reserved</td> </tr> <tr> <td>0x01</td> <td>IEEE Std 802.3</td> </tr> <tr> <td>0x02</td> <td>IEEE Std 802.9 ISLAN-16T</td> </tr> <tr> <td>0x03</td> <td>IEEE Std 802.5</td> </tr> <tr> <td>0x04</td> <td>IEEE Std 1394</td> </tr> <tr> <td>0x05 – 0x1F</td> <td>Reserved</td> </tr> </tbody> </table>	Value	Meaning	0x00	Reserved	0x01	IEEE Std 802.3	0x02	IEEE Std 802.9 ISLAN-16T	0x03	IEEE Std 802.5	0x04	IEEE Std 1394	0x05 – 0x1F	Reserved
Value	Meaning																	
0x00	Reserved																	
0x01	IEEE Std 802.3																	
0x02	IEEE Std 802.9 ISLAN-16T																	
0x03	IEEE Std 802.5																	
0x04	IEEE Std 1394																	
0x05 – 0x1F	Reserved																	

Register 23: Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6), offset 0x06

This register enables software to determine the Auto-Negotiation and Next Page capabilities of the PHY and the link partner after Auto-Negotiation.

Ethernet PHY Management Register 6 – Auto-Negotiation Expansion (MR6)

Base 0x4004.8000

Offset 0x06

Type RO, reset 0x0000

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

Bit/Field	Name	Type	Reset	Description
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	PDF	RC	0	Parallel Detection Fault When set, indicates that more than one technology has been detected at link up. This bit is cleared when read.
3	LPNPA	RO	0	Link Partner is Next Page Able When set, indicates that the link partner is Next Page Able.
2	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PRX	RC	0	New Page Received When set, indicates that a New Page has been received from the link partner and stored in the appropriate location. This bit remains set until the register is read.
0	LPANEGA	RO	0	Link Partner is Auto-Negotiation Able When set, indicates that the Link partner is Auto-Negotiation Able.

Register 24: Ethernet PHY Management Register 16 – Vendor-Specific (MR16), offset 0x10

This register enables software to configure the operation of vendor specific modes of the PHY.

Ethernet PHY Management Register 16 – Vendor-Specific (MR16)

Base 0x4004.8000

Offset 0x10

Type R/W, reset 0x0140

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RPTR	INPOL	reserved	TXHIM	SQEI	NL10	reserved				APOL	RVSPOL	reserved	PCSBP	RXCC	
Type	R/W	R/W	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	RO	RO	R/W	R/W
Reset	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	RPTR	R/W	0	<p>Repeater Mode</p> <p>When set, enables the repeater mode of operation. In this mode, full-duplex is not allowed and the Carrier Sense signal only responds to receive activity. If the PHY is configured to 10Base-T mode, the SQE test function is disabled.</p>
14	INPOL	R/W	0	<p>Interrupt Polarity</p> <p>1: Sets the polarity of the PHY interrupt to be active High.</p> <p>0: Sets the polarity of the PHY interrupt to active Low.</p> <p>Important: Because the Media Access Controller expects active Low interrupts from the PHY, this bit must always be written with a 0 to ensure proper operation.</p>
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	TXHIM	R/W	0	<p>Transmit High Impedance Mode</p> <p>When set, enables the transmitter High Impedance mode. In this mode, the TXOP and TXON transmitter pins are put into a high impedance state. The RXIP and RXIN pins remain fully functional.</p>
11	SQEI	R/W	0	<p>SQE Inhibit Testing</p> <p>When set, prohibits 10Base-T SQE testing.</p> <p>When 0, the SQE testing is performed by generating a Collision pulse following the completion of the transmission of a frame.</p>

Bit/Field	Name	Type	Reset	Description
10	NL10	R/W	0	<p>Natural Loopback Mode</p> <p>When set, enables the 10Base-T Natural Loopback mode. This causes the transmission data received by the PHY to be looped back onto the receive data path when 10Base-T mode is enabled.</p>
9:6	reserved	RO	0x05	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>
5	APOL	R/W	0	<p>Auto-Polarity Disable</p> <p>When set, disables the PHY's auto-polarity function.</p> <p>If this bit is 0, the PHY automatically inverts the received signal due to a wrong polarity connection during Auto-Negotiation if the PHY is in 10Base-T mode.</p>
4	RVSPOL	R/W	0	<p>Receive Data Polarity</p> <p>This bit indicates whether the receive data pulses are being inverted.</p> <p>If the APOL bit is 0, then the RVSPOL bit is read-only and indicates whether the auto polarity circuitry is reversing the polarity. In this case, a 1 in the RVSPOL bit indicates that the receive data is inverted while a 0 indicates that the receive data is not inverted.</p> <p>If the APOL bit is set, then the RVSPOL bit is writable and software can force the receive data to be inverted. Setting RVSPOL to 1 forces the receive data to be inverted while a 0 does not invert the receive data.</p>
3:2	reserved	RO	0	<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	PCSBP	R/W	0	<p>PCS Bypass</p> <p>When set, enables the bypass of the PCS and scrambling/descrambling functions in 100Base-TX mode. This mode is only valid when Auto-Negotiation is disabled and 100Base-T mode is enabled.</p>
0	RXCC	R/W	0	<p>Receive Clock Control</p> <p>When set, enables the Receive Clock Control power saving mode if the PHY is configured in 100Base-TX mode. This mode shuts down the receive clock when no data is being received from the physical medium to save power. This mode should not be used when PCSBP is enabled and is automatically disabled when the LOOPBK bit in the MRO register is set.</p>

Register 25: Ethernet PHY Management Register 17 – Interrupt Control/Status (MR17), offset 0x11

This register provides the means for controlling and observing the events, which trigger a PHY interrupt in the **MACRIS** register. This register can also be used in a polling mode via the MII Serial Interface as a means to observe key events within the PHY via one register address. Bits 0 through 7 are status bits, which are each set to logic 1 based on an event. These bits are cleared after the register is read. Bits 8 through 15 of this register, when set to logic 1, enable their corresponding bit in the lower byte to signal a PHY interrupt in the **MACRIS** register.

Ethernet PHY Management Register 17 – Interrupt Control/Status (MR17)

Base 0x4004.8000

Offset 0x11

Type R/W, reset 0x0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	JABBER_IE	RXER_IE	PRX_IE	PDF_IE	LPACK_IE	LSCHG_IE	RFAULT_IE	ANACOMP	JABBER_INT	RXER_INT	PRX_INT	PDF_INT	LPACK_INT	LSCHG_INT	RFAULT_INT	ANACOMP
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RC	RC	RC	RC	RC	RC	RC	RC
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	0	Software should not rely on the value of 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	JABBER_IE	R/W	0	Jabber Interrupt Enable When set, enables system interrupts when a Jabber condition is detected by the PHY.
14	RXER_IE	R/W	0	Receive Error Interrupt Enable When set, enables system interrupts when a receive error is detected by the PHY.
13	PRX_IE	R/W	0	Page Received Interrupt Enable When set, enables system interrupts when a new page is received by the PHY.
12	PDF_IE	R/W	0	Parallel Detection Fault Interrupt Enable When set, enables system interrupts when a Parallel Detection Fault is detected by the PHY.
11	LPACK_IE	R/W	0	LP Acknowledge Interrupt Enable When set, enables system interrupts when FLP bursts are received with the Acknowledge bit during Auto-Negotiation.
10	LSCHG_IE	R/W	0	Link Status Change Interrupt Enable When set, enables system interrupts when the Link Status changes from OK to FAIL.

Bit/Field	Name	Type	Reset	Description
9	RFAULT_IE	R/W	0	Remote Fault Interrupt Enable When set, enables system interrupts when a Remote Fault condition is signaled by the link partner.
8	ANEGCOMP_IE	R/W	0	Auto-Negotiation Complete Interrupt Enable When set, enables system interrupts when the Auto-Negotiation sequence has completed successfully.
7	JABBER_INT	RC	0	Jabber Event Interrupt When set, indicates that a Jabber event has been detected by the 10Base-T circuitry.
6	RXER_INT	RC	0	Receive Error Interrupt When set, indicates that a receive error has been detected by the PHY.
5	PRX_INT	RC	0	Page Receive Interrupt When set, indicates that a new page has been received from the link partner during Auto-Negotiation.
4	PDF_INT	RC	0	Parallel Detection Fault Interrupt When set, indicates that a Parallel Detection Fault has been detected by the PHY during the Auto-Negotiation process.
3	LPACK_INT	RC	0	LP Acknowledge Interrupt When set, indicates that an FLP burst has been received with the Acknowledge bit set during Auto-Negotiation.
2	LSCHG_INT	RC	0	Link Status Change Interrupt When set, indicates that the link status has changed from OK to FAIL.
1	RFAULT_INT	RC	0	Remote Fault Interrupt When set, indicates that a Remote Fault condition has been signaled by the link partner.
0	ANEGCOMP_INT	RC	0	Auto-Negotiation Complete Interrupt When set, indicates that the Auto-Negotiation sequence has completed successfully.

Register 26: Ethernet PHY Management Register 18 – Diagnostic (MR18), offset 0x12

This register enables software to diagnose the results of the previous Auto-Negotiation.

Ethernet PHY Management Register 18 – Diagnostic (MR18)

Base 0x4004.8000

Offset 0x12

Type RO, reset 0x0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			ANEGF	DPLX	RATE	RXSD	RX_LOCK	reserved							
Type	RO	RO	RO	RC	RO	RO	RO	RO	RO	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
31:13	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	ANEGF	RC	0	Auto-Negotiation Failure When set, indicates that no common technology was found during Auto-Negotiation and has failed. This bit remains set until read.
11	DPLX	RO	0	Duplex Mode When set, indicates that Full-Duplex was the highest common denominator found during the Auto-Negotiation process. Otherwise, Half-Duplex was the highest common denominator found.
10	RATE	RO	0	Rate When set, indicates that 100Base-TX was the highest common denominator found during the Auto-Negotiation process. Otherwise, 10Base-TX was the highest common denominator found.
9	RXSD	RO	0	Receive Detection When set, indicates that receive signal detection has occurred (in 100Base-TX mode) or that Manchester encoded data has been detected (in 10Base-T mode).
8	RX_LOCK	RO	0	Receive PLL Lock When set, indicates that the Receive PLL has locked onto the receive signal for the selected speed of operation (10Base-T or 100Base-TX).
7:0	reserved	RO	00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 27: Ethernet PHY Management Register 19 – Transceiver Control (MR19), offset 0x13

This register enables software to set the gain of the transmit output to compensate for transformer loss.

Ethernet PHY Management Register 19 – Transceiver Control (MR19)

Base 0x4004.8000

Offset 0x13

Type R/W, reset 0x4000

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

Bit/Field	Name	Type	Reset	Description								
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								
15:14	TXO[1:0]	R/W	1	<p>Transmit Amplitude Selection</p> <p>The TXO field sets the transmit output amplitude to account for transmit transformer insertion loss.</p> <p>Value Meaning</p> <table border="1"> <tr> <td>00</td> <td>Gain set for 0.0dB of insertion loss</td> </tr> <tr> <td>01</td> <td>Gain set for 0.4dB of insertion loss</td> </tr> <tr> <td>10</td> <td>Gain set for 0.8dB of insertion loss</td> </tr> <tr> <td>11</td> <td>Gain set for 1.2dB of insertion loss</td> </tr> </table>	00	Gain set for 0.0dB of insertion loss	01	Gain set for 0.4dB of insertion loss	10	Gain set for 0.8dB of insertion loss	11	Gain set for 1.2dB of insertion loss
00	Gain set for 0.0dB of insertion loss											
01	Gain set for 0.4dB of insertion loss											
10	Gain set for 0.8dB of insertion loss											
11	Gain set for 1.2dB of insertion loss											
13:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.								

Register 28: Ethernet PHY Management Register 23 – LED Configuration (MR23), offset 0x17

This register enables software to select the source that will cause the LEDs to toggle.

Ethernet PHY Management Register 23 – LED Configuration (MR23)

Base 0x4004.8000

Offset 0x17

Type R/W, reset 0x0010

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								LED1[3:0]				LED0[3:0]			
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	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:4	LED1[3:0]	R/W	1	The LED1 field selects the source that will toggle the LED1 signal. Value Meaning 0000 Link OK 0001 RX or TX Activity (Default LED1) 0010 TX Activity 0011 RX Activity 0100 Collision 0101 100BASE-TX mode 0110 10BASE-T mode 0111 Full Duplex 1000 Link OK & Blink=RX or TX Activity
3:0	LED0[3:0]	R/W	0	The LED0 field selects the source that will toggle the LED0 signal. Value Meaning 0000 Link OK (Default LED0) 0001 RX or TX Activity 0010 TX Activity 0011 RX Activity 0100 Collision 0101 100BASE-TX mode 0110 10BASE-T mode 0111 Full Duplex 1000 Link OK & Blink=RX or TX Activity

Register 29: Ethernet PHY Management Register 24 –MDI/MDIX Control (MR24), offset 0x18

This register enables software to control the behavior of the MDI/MDIX mux and its switching capabilities.

Ethernet PHY Management Register 24 –MDI/MDIX Control (MR24)

Base 0x4004.8000

Offset 0x18

Type R/W, reset 0x00C0

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	PD_MODE	R/W	0	Parallel Detection Mode When set, enables the Parallel Detection mode and allows auto-switching to work when Auto-Negotiation is not enabled.
6	AUTO_SW	R/W	0	Auto-Switching Enable When set, enables Auto-Switching of the MDI/MDIX mux.
5	MDIX	R/W	0	Auto-Switching Configuration When set, indicates that the MDI/MDIX mux is in the crossover (MDIX) configuration. When 0, it indicates that the mux is in the pass-through (MDI) configuration. When the AUTO_SW bit is 1, the MDIX bit is read-only. When the AUTO_SW bit is 0, the MDIX bit is read/write and can be configured manually.
4	MDIX_CM	RO	0	Auto-Switching Complete When set, indicates that the auto-switching sequence has completed. If 0, it indicates that the sequence has not completed or that auto-switching is disabled.
3:0	MDIX_SD	R/W	0	Auto-Switching Seed This field provides the initial seed for the switching algorithm. This seed directly affects the number of attempts [5,4] respectively to write bits [3:0]. A 0 sets the seed to 0x5.

18 Analog Comparators

ACMP

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

The LM3S8938 controller provides three independent integrated analog comparators that can be configured to drive an output or generate an interrupt or ADC event.

Note: Not all comparators have the option to drive an output pin. See the Comparator Operating Mode tables for more information.

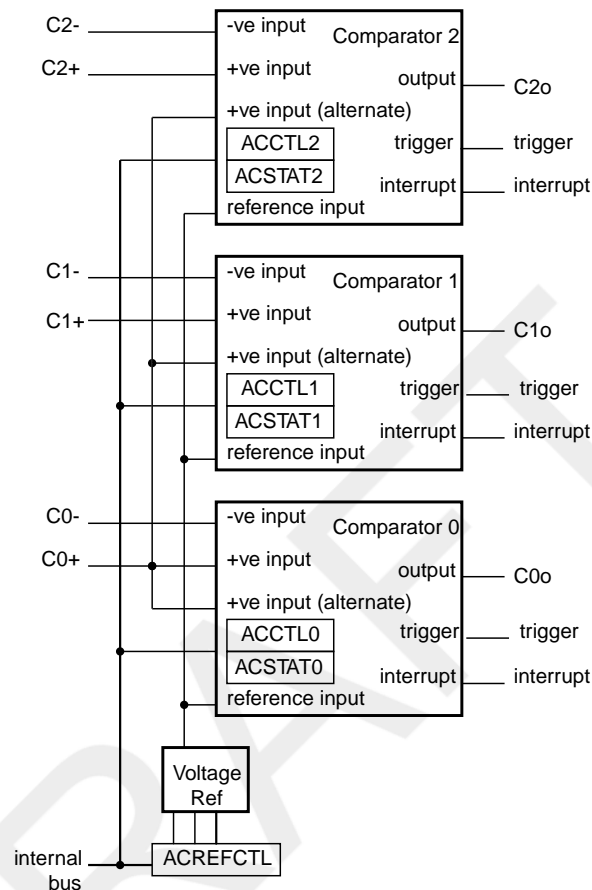
A comparator can compare a test voltage against any one of these voltages:

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

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

18.1 Block Diagram

Figure 18-1. Analog Comparator Module Block Diagram



18.2 Functional Description

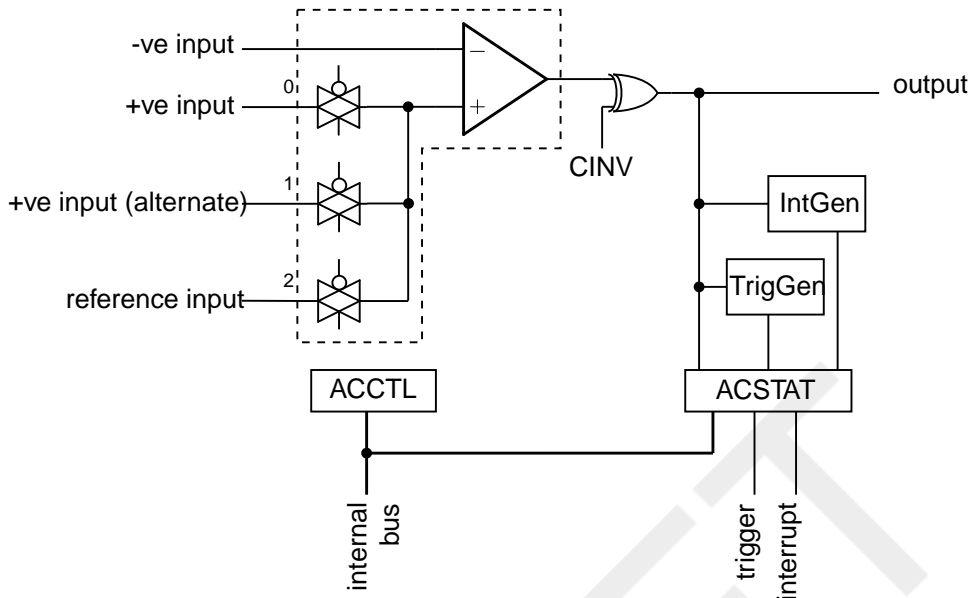
Important: It is recommended that the Digital-Input enable (the `GPIOEN` bit in the GPIO module) for the analog input pin be disabled to prevent excessive current draw from the I/O pads.

The comparator compares the V_{IN-} and V_{IN+} inputs to produce an output, V_{OUT} .

$$\begin{aligned} V_{IN-} < V_{IN+}, & \quad V_{OUT} = 1 \\ V_{IN-} > V_{IN+}, & \quad V_{OUT} = 0 \end{aligned}$$

As shown in Figure 18-2 on page 484, the input source for V_{IN-} is an external input. In addition to an external input, input sources for V_{IN+} can be the +ve input of comparator 0 or an internal reference.

Figure 18-2. Structure of Comparator Unit



A comparator is configured through two status/control registers (**ACCTL** and **ACSTAT**). The internal reference is configured through one control register (**ACREFCTL**). Interrupt status and control is configured through three registers (**ACMIS**, **ACRIS**, and **ACINTEN**). The operating modes of the comparators are shown in the Comparator Operating Mode tables.

Typically, the comparator output is used internally to generate controller interrupts. It may also be used to drive an external pin or generate an analog-to-digital converter (ADC) trigger.

Important: Certain register bit values must be set before using the analog comparators. The proper pad configuration for the comparator input and output pins are described in the Comparator Operating Mode tables.

Table 18-1. Comparator 0 Operating Modes

ACCNTL0 Comparator 0					
ASRCP	VIN-	VIN+	Output	Interrupt	ADC Trigger
00	C0-	C0+	C0o	yes	yes
01	C0-	C0+	C0o	yes	yes
10	C0-	Vref	C0o	yes	yes
11	C0-	reserved	C0o	yes	yes

Table 18-2. Comparator 1 Operating Modes

ACCNTL1 Comparator 1					
ASRCP	VIN-	VIN+	Output	Interrupt	ADC Trigger
00	C1-	C1o/C1+ ^a	C1o/C1+	yes	yes
01	C1-	C0+	C1o/C1+	yes	yes
10	C1-	Vref	C1o/C1+	yes	yes
11	C1-	reserved	C1o/C1+	yes	yes

a. C1o and C1+ signals share a single pin and may only be used as one or the other.

Table 18-3. Comparator 2 Operating Modes

ACCNTL2	Comparator 2				
ASRCP	VIN-	VIN+	Output	Interrupt	ADC Trigger
00	C2-	C2o/C2+ ^a	C2o/C2+	yes	yes
01	C2-	C0+	C2o/C2+	yes	yes
10	C2-	Vref	C2o/C2+	yes	yes
11	C2-	reserved	C2o/C2+	yes	yes

a. C2o and C2+ signals share a single pin and may only be used as one or the other.

18.2.1 Internal Reference Programming

The structure of the internal reference is shown in Figure 18-3 on page 485. This is controlled by a single configuration register (**ACREFCTL**). Table 18-4 on page 485 shows the programming options to develop specific internal reference values, to compare an external voltage against a particular voltage generated internally.

Figure 18-3. Comparator Internal Reference Structure

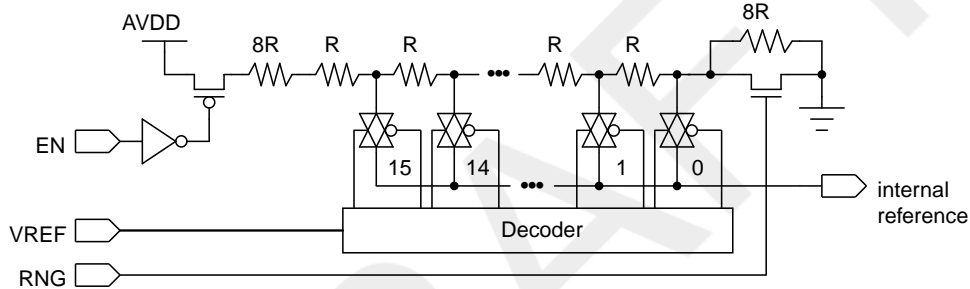


Table 18-4. Internal Reference Voltage and ACREFTL Field Values

ACREFCTL Register		Output Reference Voltage Based on VREF Field Value
EN Bit Value	RNG Bit Value	
EN=0	RNG=X	0 V (GND) for any value of VREF; however, it is recommended that RNG=1 and VREF=0 for the least noisy ground reference.

ACREFCTL Register		Output Reference Voltage Based on VREF Field Value
EN Bit Value	RNG Bit Value	
EN=1	RNG=0	<p>Total resistance in ladder is 32 R.</p> $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF + 8)}{32}$ $V_{REF} = 0.825 + 0.103 VREF$ <p>The range of internal reference in this mode is 0.825-2.37 V.</p>
	RNG=1	<p>Total resistance in ladder is 24 R.</p> $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF)}{24}$ $V_{REF} = 0.1375 \times VREF$ <p>The range of internal reference for this mode is 0.0-2.0625 V.</p>

18.3 Initialization and Configuration

The following example shows how to configure an analog comparator to read back its output value from an internal register.

1. Enable the analog comparator 0 clock by writing a value of 0x0010.0000 to the **RCGC1** register in the System Control module.
2. In the GPIO module, enable the GPIO port/pin associated with C0- as a GPIO input.
3. Configure the internal voltage reference to 1.65 V by writing the **ACREFCTL** register with the value 0x0000.030C.
4. Configure comparator 0 to use the internal voltage reference and to *not* invert the output on the C0o pin by writing the **ACCTL0** register with the value of 0x0000.040C.
5. Delay for some time.
6. Read the comparator output value by reading the **ACSTAT0** register's OVAL value.

Change the level of the signal input on C0- to see the OVAL value change.

18.4 Register Map

Table 18-5 on page 487 lists the comparator registers. The offset listed is a hexadecimal increment to the register's address, relative to the Analog Comparator base address of 0x4003.C000.

Table 18-5. Analog Comparators Register Map

Offset	Name	Type	Reset	Description	See page
0x00	ACMIS	R/W1C	0x0000.0000	Analog Comparator Masked Interrupt Status	488
0x04	ACRIS	RO	0x0000.0000	Analog Comparator Raw Interrupt Status	489
0x08	ACINTEN	R/W	0x0000.0000	Analog Comparator Interrupt Enable	490
0x10	ACREFCTL	R/W	0x0000.0000	Analog Comparator Reference Voltage Control	491
0x20	ACSTAT0	RO	0x0000.0000	Analog Comparator Status 0	492
0x24	ACCTL0	R/W	0x0000.0000	Analog Comparator Control 0	493
0x40	ACSTAT1	RO	0x0000.0000	Analog Comparator Status 1	492
0x44	ACCTL1	R/W	0x0000.0000	Analog Comparator Control 1	493
0x60	ACSTAT2	RO	0x0000.0000	Analog Comparator Status 2	492
0x64	ACCTL2	R/W	0x0000.0000	Analog Comparator Control 2	493

18.5 Register Descriptions

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

Register 1: Analog Comparator Masked Interrupt Status (ACMIS), offset 0x00

This register provides a summary of the interrupt status (masked) of the comparator.

Analog Comparator Masked Interrupt Status (ACMIS)

Base 0x4003.C000

Offset 0x00

Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													IN2	IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
31:3	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	IN2	R/W1C	0	Comparator 2 Masked Interrupt Status Gives the masked interrupt state of this interrupt. Write 1 to this bit to clear the pending interrupt.
1	IN1	R/W1C	0	Comparator 1 Masked Interrupt Status Gives the masked interrupt state of this interrupt. Write 1 to this bit to clear the pending interrupt.
0	IN0	R/W1C	0	Comparator 0 Masked Interrupt Status Gives the masked interrupt state of this interrupt. Write 1 to this bit to clear the pending interrupt.

Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x04

This register provides a summary of the interrupt status (raw) of the comparator.

Analog Comparator Raw Interrupt Status (ACRIS)

Base 0x4003.C000

Offset 0x04

Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													IN2	IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	IN2	RO	0	When set, indicates that an interrupt has been generated by comparator 2.
1	IN1	RO	0	When set, indicates that an interrupt has been generated by comparator 1.
0	IN0	RO	0	When set, indicates that an interrupt has been generated by comparator 0.

Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x08

This register provides the interrupt enable for the comparator.

Analog Comparator Interrupt Enable (ACINTEN)

Base 0x4003.C000
 Offset 0x08
 Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													IN2	IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	IN2	R/W	0	When set, enables the controller interrupt from the comparator 2 output
1	IN1	R/W	0	When set, enables the controller interrupt from the comparator 1 output.
0	IN0	R/W	0	When set, enables the controller interrupt from the comparator 0 output.

Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x10

This register specifies whether the resistor ladder is powered on as well as the range and tap.

Analog Comparator Reference Voltage Control (ACREFCTL)

Base 0x4003.C000

Offset 0x10

Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0	Software should not rely on the value of 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	The EN bit specifies whether the resistor ladder is powered on. If 0, the resistor ladder is unpowered. If 1, the resistor ladder is connected to the analog V_{DD} . This bit is reset to 0 so that the internal reference consumes the least amount of power if not used and programmed.
8	RNG	R/W	0	The RNG bit specifies the range of the resistor ladder. If 0, the resistor ladder has a total resistance of 32 R. If 1, the resistor ladder has a total resistance of 24 R.
7:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	VREF	R/W	0	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 18-4 on page 485 for some output reference voltage examples.

Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x20

Register 6: Analog Comparator Status 1 (ACSTAT1), offset 0x40

Register 7: Analog Comparator Status 2 (ACSTAT2), offset 0x60

These registers specify the current output value of the comparator.

Analog Comparator Status 0 (ACSTAT0)

Base 0x4003.C000
 Offset 0x20
 Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of 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	The OVAL bit specifies the current output value of the comparator.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 8: Analog Comparator Control 0 (ACCTL0), offset 0x24**Register 9: Analog Comparator Control 1 (ACCTL1), offset 0x44****Register 10: Analog Comparator Control 2 (ACCTL2), offset 0x64**

These registers configure the comparator's input and output.

Analog Comparator Control 0 (ACCTL0)

Base 0x4003.C000

Offset 0x24

Type R/W, reset 0x0000.0000

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

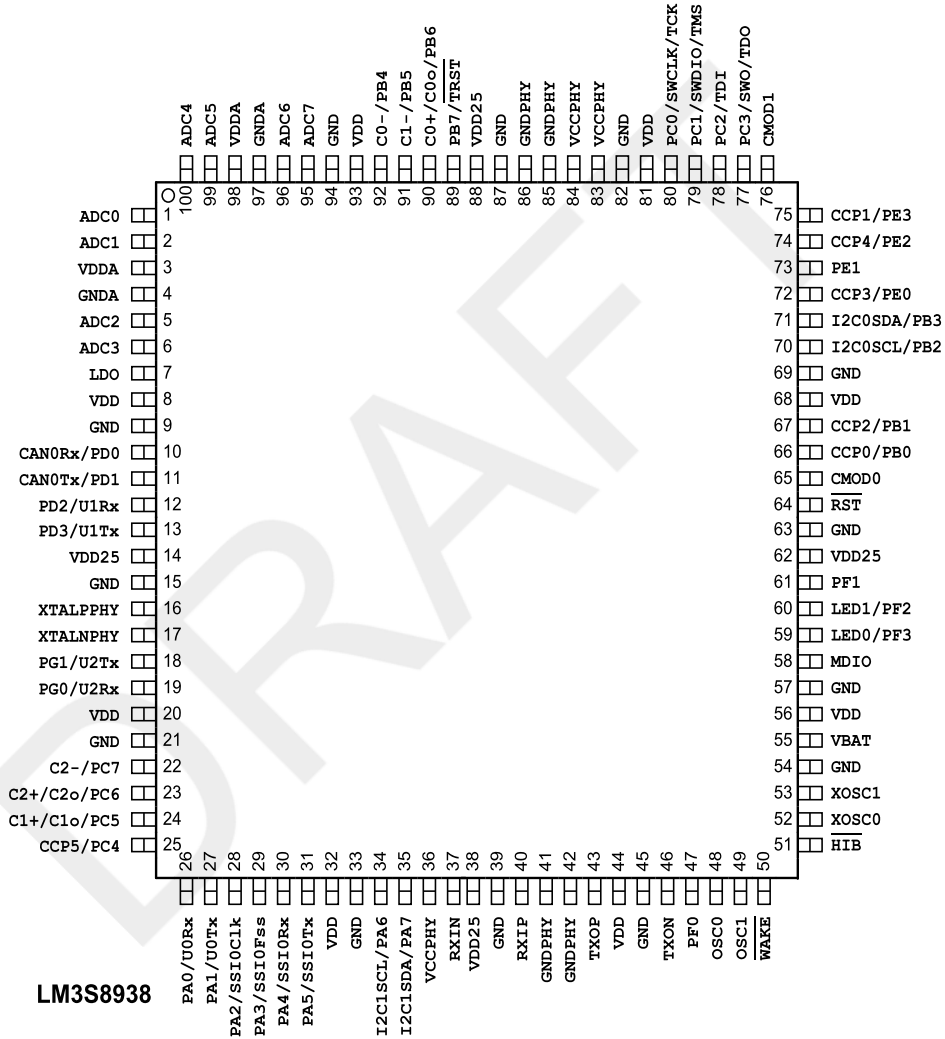
Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TOEN	R/W	0	The TOEN bit enables the ADC event transmission to the ADC. If 0, the event is suppressed and not sent to the ADC. If 1, the event is transmitted to the ADC.
10:9	ASRCP	R/W	0	The ASRCP field specifies the source of input voltage to the VIN+ terminal of the comparator. The encodings for this field are as follows: ASRCP Function 00 Pin value 01 Pin value of C0+ 10 Internal voltage reference 11 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	The TSLVAL bit specifies the sense value of the input that generates an ADC event if in Level Sense mode. If 0, an ADC event is generated if the comparator output is Low. Otherwise, an ADC event is generated if the comparator output is High.

Bit/Field	Name	Type	Reset	Description								
6:5	TSEN	R/W	0	<p>The TSEN field specifies the sense of the comparator output that generates an ADC event. The sense conditioning is as follows:</p> <p>TSEN Function</p> <table border="0"> <tr> <td>00</td> <td>Level sense, see TSLVAL</td> </tr> <tr> <td>01</td> <td>Falling edge</td> </tr> <tr> <td>10</td> <td>Rising edge</td> </tr> <tr> <td>11</td> <td>Either edge</td> </tr> </table>	00	Level sense, see TSLVAL	01	Falling edge	10	Rising edge	11	Either edge
00	Level sense, see TSLVAL											
01	Falling edge											
10	Rising edge											
11	Either edge											
4	ISLVAL	R/W	0	<p>The ISLVAL bit specifies the sense value of the input that generates an interrupt if in Level Sense mode. If 0, an interrupt is generated if the comparator output is Low. Otherwise, an interrupt is generated if the comparator output is High.</p>								
3:2	ISEN	R/W	0	<p>The ISEN field specifies the sense of the comparator output that generates an interrupt. The sense conditioning is as follows:</p> <p>ISEN Function</p> <table border="0"> <tr> <td>00</td> <td>Level sense, see ISLVAL</td> </tr> <tr> <td>01</td> <td>Falling edge</td> </tr> <tr> <td>10</td> <td>Rising edge</td> </tr> <tr> <td>11</td> <td>Either edge</td> </tr> </table>	00	Level sense, see ISLVAL	01	Falling edge	10	Rising edge	11	Either edge
00	Level sense, see ISLVAL											
01	Falling edge											
10	Rising edge											
11	Either edge											
1	CINV	R/W	0	<p>The CINV bit conditionally inverts the output of the comparator. If 0, the output of the comparator is unchanged. If 1, the output of the comparator is inverted prior to being processed by hardware.</p>								
0	reserved	RO	0	<p>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</p>								

19 Pin Diagram

Figure 19-1 on page 495 shows the pin diagram and pin-to-signal-name mapping.

Figure 19-1. Pin Connection Diagram



20 Signal Tables

The following tables list the signals available for each pin. Functionality is enabled by software with the GPIOAFSEL register.

Important: All multiplexed pins are GPIOs by default, with the exception of the five JTAG pins (PB7 and PC[3:0]) which default to the JTAG functionality.

Table 20-1 on page 496 shows the pin-to-signal-name mapping, including functional characteristics of the signals. Table 20-2 on page 500 lists the signals in alphabetical order by signal name.

Table 20-3 on page 504 groups the signals by functionality, except for GPIOs. Table 20-4 on page 508 lists the GPIO pins and their alternate functionality.

Table 20-1. Signals by Pin Number

Pin Number	Pin Name	Pin Type	Buffer Type	Description
1	ADC0	I	Analog	Analog-to-digital converter input 0.
2	ADC1	I	Analog	Analog-to-digital converter input 1.
3	VDDA	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
4	GNDA	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
5	ADC2	I	Analog	Analog-to-digital converter input 2.
6	ADC3	I	Analog	Analog-to-digital converter input 3.
7	LDO	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 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 VDD25 pins at the board level in addition to the decoupling capacitor(s).
8	VDD	-	Power	Positive supply for I/O and some logic.
9	GND	-	Power	Ground reference for logic and I/O pins.
10	CAN0Rx	I	TTL	CAN module 0 receive
	PD0	I/O	TTL	GPIO port D bit 0
11	CAN0Tx	O	TTL	CAN module 0 transmit
	PD1	I/O	TTL	GPIO port D bit 1
12	PD2	I/O	TTL	GPIO port D bit 2
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
13	PD3	I/O	TTL	GPIO port D bit 3
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.

Pin Number	Pin Name	Pin Type	Buffer Type	Description
14	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
15	GND	-	Power	Ground reference for logic and I/O pins.
16	XTALPPHY	O	TTL	XTALP of the Ethernet PHY
17	XTALNPHY	I	TTL	XTALN of the Ethernet PHY
18	PG1	I/O	TTL	GPIO port G bit 1
	U2Tx	O	TTL	UART 2 Transmit. When in IrDA mode, this signal has IrDA modulation.
19	PG0	I/O	TTL	GPIO port G bit 0
	U2Rx	I	TTL	UART 2 Receive. When in IrDA mode, this signal has IrDA modulation.
20	VDD	-	Power	Positive supply for I/O and some logic.
21	GND	-	Power	Ground reference for logic and I/O pins.
22	C2-	I	Analog	Analog comparator 2 negative input
	PC7	I/O	TTL	GPIO port C bit 7
23	C2+	I	Analog	Analog comparator positive input
	C2o	O	TTL	Analog comparator 2 output
	PC6	I/O	TTL	GPIO port C bit 6
24	C1+	I	Analog	Analog comparator positive input
	C1o	O	TTL	Analog comparator 1 output
	PC5	I/O	TTL	GPIO port C bit 5
25	CCP5	I/O	TTL	Capture/Compare/PWM 5
	PC4	I/O	TTL	GPIO port C bit 4
26	PA0	I/O	TTL	GPIO port A bit 0
	U0Rx	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
27	PA1	I/O	TTL	GPIO port A bit 1
	U0Tx	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
28	PA2	I/O	TTL	GPIO port A bit 2
	SSI0Clk	I/O	TTL	SSI module 0 clock
29	PA3	I/O	TTL	GPIO port A bit 3
	SSI0Fss	I/O	TTL	SSI module 0 frame
30	PA4	I/O	TTL	GPIO port A bit 4
	SSI0Rx	I	TTL	SSI module 0 receive
31	PA5	I/O	TTL	GPIO port A bit 5
	SSI0Tx	O	TTL	SSI module 0 transmit
32	VDD	-	Power	Positive supply for I/O and some logic.
33	GND	-	Power	Ground reference for logic and I/O pins.
34	I2C1SCL	I/O	OD	I2C module 1 clock
	PA6	I/O	TTL	GPIO port A bit 6
35	I2C1SDA	I/O	OD	I2C module 1 data
	PA7	I/O	TTL	GPIO port A bit 7

Pin Number	Pin Name	Pin Type	Buffer Type	Description
36	VCCPHY	I	TTL	VCC of the Ethernet PHY
37	RXIN	I	Analog	RXIN of the Ethernet PHY
38	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
39	GND	-	Power	Ground reference for logic and I/O pins.
40	RXIP	I	Analog	RXIP of the Ethernet PHY
41	GNDPHY	I	TTL	GND of the Ethernet PHY
42	GNDPHY	I	TTL	GND of Ethernet PHY
43	TXOP	O	Analog	TXOP of Ethernet PHY
44	VDD	-	Power	Positive supply for I/O and some logic.
45	GND	-	Power	Ground reference for logic and I/O pins.
46	TXON	O	Analog	TXON of Ethernet PHY
47	PF0	I/O	TTL	GPIO port F bit 0
48	OSC0	I	Analog	Main oscillator crystal input or an external clock reference input.
49	OSC1	O	Analog	Main oscillator crystal output.
50	$\overline{\text{WAKE}}$	I	OD	An external input that brings the processor out of hibernate mode when asserted.
51	$\overline{\text{HIB}}$	O	TTL	An output that indicates the processor is in hibernate mode.
52	XOSC0	I	Analog	Hibernation Module oscillator crystal input or an external clock reference input. Note that this is either a 4.19-MHz crystal or a 32.768-kHz oscillator for the Hibernation Module RTC. See the CLKSEL bit in the HIBCTL register.
53	XOSC1	O	Analog	Hibernation Module oscillator crystal output.
54	GND	-	Power	Ground reference for logic and I/O pins.
55	VBAT	-	Power	Power source for the Hibernation Module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation Module power-source supply.
56	VDD	-	Power	Positive supply for I/O and some logic.
57	GND	-	Power	Ground reference for logic and I/O pins.
58	MDIO	I/O	TTL	MDIO of Ethernet PHY
59	LED0	O	TTL	MII LED 0
	PF3	I/O	TTL	GPIO port F bit 3
60	LED1	O	TTL	MII LED 1
	PF2	I/O	TTL	GPIO port F bit 2
61	PF1	I/O	TTL	GPIO port F bit 1
62	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
63	GND	-	Power	Ground reference for logic and I/O pins.
64	$\overline{\text{RST}}$	I	TTL	System reset input.

Pin Number	Pin Name	Pin Type	Buffer Type	Description
65	CMOD0	I/O	TTL	CPU Mode bit 0. Input must be set to logic 0 (grounded); other encodings reserved.
66	CCP0	I/O	TTL	Capture/Compare/PWM 0
	PB0	I/O	TTL	GPIO port B bit 0
67	CCP2	I/O	TTL	Capture/Compare/PWM 2
	PB1	I/O	TTL	GPIO port B bit 1
68	VDD	-	Power	Positive supply for I/O and some logic.
69	GND	-	Power	Ground reference for logic and I/O pins.
70	I2C0SCL	I/O	OD	I2C module 0 clock
	PB2	I/O	TTL	GPIO port B bit 2
71	I2C0SDA	I/O	OD	I2C module 0 data
	PB3	I/O	TTL	GPIO port B bit 3
72	CCP3	I/O	TTL	Capture/Compare/PWM 3
	PE0	I/O	TTL	GPIO port E bit 0
73	PE1	I/O	TTL	GPIO port E bit 1
74	CCP4	I/O	TTL	Capture/Compare/PWM 4
	PE2	I/O	TTL	GPIO port E bit 2
75	CCP1	I/O	TTL	Capture/Compare/PWM 1
	PE3	I/O	TTL	GPIO port E bit 3
76	CMOD1	I/O	TTL	CPU Mode bit 1. Input must be set to logic 0 (grounded); other encodings reserved.
77	PC3	I/O	TTL	GPIO port C bit 3
	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/O	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	VCCPHY	I	TTL	VCC of the Ethernet PHY
84	VCCPHY	I	TTL	VCC of the Ethernet PHY
85	GNDPHY	I	TTL	GND of the Ethernet PHY
86	GNDPHY	I	TTL	GND of the Ethernet PHY
87	GND	-	Power	Ground reference for logic and I/O pins.
88	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
89	PB7	I/O	TTL	GPIO port B bit 7
	TRST	I	TTL	JTAG TRSTn

Pin Number	Pin Name	Pin Type	Buffer Type	Description
90	C0+	I	Analog	Analog comparator 0 positive input
	C0o	O	TTL	Analog comparator 0 output
	PB6	I/O	TTL	GPIO port B bit 6
91	C1-	I	Analog	Analog comparator 1 negative input
	PB5	I/O	TTL	GPIO port B bit 5
92	C0-	I	Analog	Analog comparator 0 negative input
	PB4	I/O	TTL	GPIO port B bit 4
93	VDD	-	Power	Positive supply for I/O and some logic.
94	GND	-	Power	Ground reference for logic and I/O pins.
95	ADC7	I	Analog	Analog-to-digital converter input 7.
96	ADC6	I	Analog	Analog-to-digital converter input 6.
97	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.
98	VDD _A	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
99	ADC5	I	Analog	Analog-to-digital converter input 5.
100	ADC4	I	Analog	Analog-to-digital converter input 4.

Table 20-2. Signals by Signal Name

Pin Name	Pin Number	Pin Type	Buffer Type	Description
ADC0	1	I	Analog	Analog-to-digital converter input 0.
ADC1	2	I	Analog	Analog-to-digital converter input 1.
ADC2	5	I	Analog	Analog-to-digital converter input 2.
ADC3	6	I	Analog	Analog-to-digital converter input 3.
ADC4	100	I	Analog	Analog-to-digital converter input 4.
ADC5	99	I	Analog	Analog-to-digital converter input 5.
ADC6	96	I	Analog	Analog-to-digital converter input 6.
ADC7	95	I	Analog	Analog-to-digital converter input 7.
C0+	90	I	Analog	Analog comparator 0 positive input
C0-	92	I	Analog	Analog comparator 0 negative input
C0o	90	O	TTL	Analog comparator 0 output
C1+	24	I	Analog	Analog comparator positive input
C1-	91	I	Analog	Analog comparator 1 negative input
C1o	24	O	TTL	Analog comparator 1 output
C2+	23	I	Analog	Analog comparator positive input
C2-	22	I	Analog	Analog comparator 2 negative input
C2o	23	O	TTL	Analog comparator 2 output
CAN0Rx	10	I	TTL	CAN module 0 receive
CAN0Tx	11	O	TTL	CAN module 0 transmit

Pin Name	Pin Number	Pin Type	Buffer Type	Description
CCP0	66	I/O	TTL	Capture/Compare/PWM 0
CCP1	75	I/O	TTL	Capture/Compare/PWM 1
CCP2	67	I/O	TTL	Capture/Compare/PWM 2
CCP3	72	I/O	TTL	Capture/Compare/PWM 3
CCP4	74	I/O	TTL	Capture/Compare/PWM 4
CCP5	25	I/O	TTL	Capture/Compare/PWM 5
CMOD0	65	I/O	TTL	CPU Mode bit 0. Input must be set to logic 0 (grounded); other encodings reserved.
CMOD1	76	I/O	TTL	CPU Mode bit 1. Input must be set to logic 0 (grounded); other encodings reserved.
GND	9	-	Power	Ground reference for logic and I/O pins.
GND	15	-	Power	Ground reference for logic and I/O pins.
GND	21	-	Power	Ground reference for logic and I/O pins.
GND	33	-	Power	Ground reference for logic and I/O pins.
GND	39	-	Power	Ground reference for logic and I/O pins.
GND	45	-	Power	Ground reference for logic and I/O pins.
GND	54	-	Power	Ground reference for logic and I/O pins.
GND	57	-	Power	Ground reference for logic and I/O pins.
GND	63	-	Power	Ground reference for logic and I/O pins.
GND	69	-	Power	Ground reference for logic and I/O pins.
GND	82	-	Power	Ground reference for logic and I/O pins.
GND	87	-	Power	Ground reference for logic and I/O pins.
GND	94	-	Power	Ground reference for logic and I/O pins.
GND _A	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.
GND _A	97	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
GND _{PHY}	41	I	TTL	GND of the Ethernet PHY
GND _{PHY}	42	I	TTL	GND of Ethernet PHY
GND _{PHY}	85	I	TTL	GND of the Ethernet PHY
GND _{PHY}	86	I	TTL	GND of the Ethernet PHY
HIB	51	O	TTL	An output that indicates the processor is in hibernate mode.
I2C0SCL	70	I/O	OD	I2C module 0 clock
I2C0SDA	71	I/O	OD	I2C module 0 data
I2C1SCL	34	I/O	OD	I2C module 1 clock
I2C1SDA	35	I/O	OD	I2C module 1 data

Pin Name	Pin Number	Pin Type	Buffer Type	Description
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 VDD25 pins at the board level in addition to the decoupling capacitor(s).
LED0	59	O	TTL	MII LED 0
LED1	60	O	TTL	MII LED 1
MDIO	58	I/O	TTL	MDIO of Ethernet PHY
OSC0	48	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	49	O	Analog	Main oscillator crystal output.
PA0	26	I/O	TTL	GPIO port A bit 0
PA1	27	I/O	TTL	GPIO port A bit 1
PA2	28	I/O	TTL	GPIO port A bit 2
PA3	29	I/O	TTL	GPIO port A bit 3
PA4	30	I/O	TTL	GPIO port A bit 4
PA5	31	I/O	TTL	GPIO port A bit 5
PA6	34	I/O	TTL	GPIO port A bit 6
PA7	35	I/O	TTL	GPIO port A bit 7
PB0	66	I/O	TTL	GPIO port B bit 0
PB1	67	I/O	TTL	GPIO port B bit 1
PB2	70	I/O	TTL	GPIO port B bit 2
PB3	71	I/O	TTL	GPIO port B bit 3
PB4	92	I/O	TTL	GPIO port B bit 4
PB5	91	I/O	TTL	GPIO port B bit 5
PB6	90	I/O	TTL	GPIO port B bit 6
PB7	89	I/O	TTL	GPIO port B bit 7
PC0	80	I/O	TTL	GPIO port C bit 0
PC1	79	I/O	TTL	GPIO port C bit 1
PC2	78	I/O	TTL	GPIO port C bit 2
PC3	77	I/O	TTL	GPIO port C bit 3
PC4	25	I/O	TTL	GPIO port C bit 4
PC5	24	I/O	TTL	GPIO port C bit 5
PC6	23	I/O	TTL	GPIO port C bit 6
PC7	22	I/O	TTL	GPIO port C bit 7
PD0	10	I/O	TTL	GPIO port D bit 0
PD1	11	I/O	TTL	GPIO port D bit 1
PD2	12	I/O	TTL	GPIO port D bit 2
PD3	13	I/O	TTL	GPIO port D bit 3
PE0	72	I/O	TTL	GPIO port E bit 0
PE1	73	I/O	TTL	GPIO port E bit 1
PE2	74	I/O	TTL	GPIO port E bit 2
PE3	75	I/O	TTL	GPIO port E bit 3

Pin Name	Pin Number	Pin Type	Buffer Type	Description
PF0	47	I/O	TTL	GPIO port F bit 0
PF1	61	I/O	TTL	GPIO port F bit 1
PF2	60	I/O	TTL	GPIO port F bit 2
PF3	59	I/O	TTL	GPIO port F bit 3
PG0	19	I/O	TTL	GPIO port G bit 0
PG1	18	I/O	TTL	GPIO port G bit 1
$\overline{\text{RST}}$	64	I	TTL	System reset input.
RXIN	37	I	Analog	RXIN of the Ethernet PHY
RXIP	40	I	Analog	RXIP of the Ethernet PHY
SSI0Clk	28	I/O	TTL	SSI module 0 clock
SSI0Fss	29	I/O	TTL	SSI module 0 frame
SSI0Rx	30	I	TTL	SSI module 0 receive
SSI0Tx	31	O	TTL	SSI module 0 transmit
SWCLK	80	I	TTL	JTAG/SWD CLK
SWDIO	79	I/O	TTL	JTAG TMS and SWDIO
SWO	77	O	TTL	JTAG TDO and SWO
TCK	80	I	TTL	JTAG/SWD CLK
TDI	78	I	TTL	JTAG TDI
TDO	77	O	TTL	JTAG TDO and SWO
TMS	79	I/O	TTL	JTAG TMS and SWDIO
$\overline{\text{TRST}}$	89	I	TTL	JTAG TRSTn
TXON	46	O	Analog	TXON of Ethernet PHY
TXOP	43	O	Analog	TXOP of Ethernet PHY
U0Rx	26	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
U0Tx	27	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
U1Rx	12	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
U1Tx	13	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
U2Rx	19	I	TTL	UART 2 Receive. When in IrDA mode, this signal has IrDA modulation.
U2Tx	18	O	TTL	UART 2 Transmit. When in IrDA mode, this signal has IrDA modulation.
VBAT	55	-	Power	Power source for the Hibernation Module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation Module power-source supply.
VCCPHY	36	I	TTL	VCC of the Ethernet PHY
VCCPHY	83	I	TTL	VCC of the Ethernet PHY
VCCPHY	84	I	TTL	VCC of the Ethernet PHY
VDD	8	-	Power	Positive supply for I/O and some logic.
VDD	20	-	Power	Positive supply for I/O and some logic.
VDD	32	-	Power	Positive supply for I/O and some logic.

Pin Name	Pin Number	Pin Type	Buffer Type	Description
VDD	44	-	Power	Positive supply for I/O and some logic.
VDD	56	-	Power	Positive supply for I/O and some logic.
VDD	68	-	Power	Positive supply for I/O and some logic.
VDD	81	-	Power	Positive supply for I/O and some logic.
VDD	93	-	Power	Positive supply for I/O and some logic.
VDD25	14	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDD25	38	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDD25	62	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDD25	88	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDDA	3	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
VDDA	98	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
$\overline{\text{WAKE}}$	50	I	OD	An external input that brings the processor out of hibernate mode when asserted.
XOSC0	52	I	Analog	Hibernation Module oscillator crystal input or an external clock reference input. Note that this is either a 4.19-MHz crystal or a 32.768-kHz oscillator for the Hibernation Module RTC. See the CLKSEL bit in the HIBCTL register.
XOSC1	53	O	Analog	Hibernation Module oscillator crystal output.
XTALNPHY	17	I	TTL	XTALN of the Ethernet PHY
XTALPPHY	16	O	TTL	XTALP of the Ethernet PHY

Table 20-3. Signals by Function, Except for GPIO

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
ADC	ADC0	1	I	Analog	Analog-to-digital converter input 0.
	ADC1	2	I	Analog	Analog-to-digital converter input 1.
	ADC2	5	I	Analog	Analog-to-digital converter input 2.
	ADC3	6	I	Analog	Analog-to-digital converter input 3.
	ADC4	100	I	Analog	Analog-to-digital converter input 4.
	ADC5	99	I	Analog	Analog-to-digital converter input 5.
	ADC6	96	I	Analog	Analog-to-digital converter input 6.
	ADC7	95	I	Analog	Analog-to-digital converter input 7.

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
Analog Comparators	C0+	90	I	Analog	Analog comparator 0 positive input
	C0-	92	I	Analog	Analog comparator 0 negative input
	C0o	90	O	TTL	Analog comparator 0 output
	C1+	24	I	Analog	Analog comparator positive input
	C1-	91	I	Analog	Analog comparator 1 negative input
	C1o	24	O	TTL	Analog comparator 1 output
	C2+	23	I	Analog	Analog comparator positive input
	C2-	22	I	Analog	Analog comparator 2 negative input
	C2o	23	O	TTL	Analog comparator 2 output
Controller Area Network	CANORx	10	I	TTL	CAN module 0 receive
	CANOTx	11	O	TTL	CAN module 0 transmit
Ethernet PHY	GNDPHY	41	I	TTL	GND of the Ethernet PHY
	GNDPHY	42	I	TTL	GND of Ethernet PHY
	GNDPHY	85	I	TTL	GND of the Ethernet PHY
	GNDPHY	86	I	TTL	GND of the Ethernet PHY
	LED0	59	O	TTL	MII LED 0
	LED1	60	O	TTL	MII LED 1
	MDIO	58	I/O	TTL	MDIO of Ethernet PHY
	RXIN	37	I	Analog	RXIN of the Ethernet PHY
	RXIP	40	I	Analog	RXIP of the Ethernet PHY
	TXON	46	O	Analog	TXON of Ethernet PHY
	TXOP	43	O	Analog	TXOP of Ethernet PHY
	VCCPHY	36	I	TTL	VCC of the Ethernet PHY
	VCCPHY	83	I	TTL	VCC of the Ethernet PHY
	VCCPHY	84	I	TTL	VCC of the Ethernet PHY
	XTALNPHY	17	I	TTL	XTALN of the Ethernet PHY
	XTALPPHY	16	O	TTL	XTALP of the Ethernet PHY
General-Purpose Timers	CCP0	66	I/O	TTL	Capture/Compare/PWM 0
	CCP1	75	I/O	TTL	Capture/Compare/PWM 1
	CCP2	67	I/O	TTL	Capture/Compare/PWM 2
	CCP3	72	I/O	TTL	Capture/Compare/PWM 3
	CCP4	74	I/O	TTL	Capture/Compare/PWM 4
	CCP5	25	I/O	TTL	Capture/Compare/PWM 5
I2C	I2C0SCL	70	I/O	OD	I2C module 0 clock
	I2C0SDA	71	I/O	OD	I2C module 0 data
	I2C1SCL	34	I/O	OD	I2C module 1 clock
	I2C1SDA	35	I/O	OD	I2C module 1 data

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
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/O	TTL	JTAG TMS and SWDIO

DRAFT

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
Power	GND	9	-	Power	Ground reference for logic and I/O pins.
	GND	15	-	Power	Ground reference for logic and I/O pins.
	GND	21	-	Power	Ground reference for logic and I/O pins.
	GND	33	-	Power	Ground reference for logic and I/O pins.
	GND	39	-	Power	Ground reference for logic and I/O pins.
	GND	45	-	Power	Ground reference for logic and I/O pins.
	GND	54	-	Power	Ground reference for logic and I/O pins.
	GND	57	-	Power	Ground reference for logic and I/O pins.
	GND	63	-	Power	Ground reference for logic and I/O pins.
	GND	69	-	Power	Ground reference for logic and I/O pins.
	GND	82	-	Power	Ground reference for logic and I/O pins.
	GND	87	-	Power	Ground reference for logic and I/O pins.
	GND	94	-	Power	Ground reference for logic and I/O pins.
	GND	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.
	GND	97	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
	HIB	51	O	TTL	An output that indicates the processor is in hibernate mode.
	LDO	7	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 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 VDD25 pins at the board level in addition to the decoupling capacitor(s).
	VBAT	55	-	Power	Power source for the Hibernation Module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation Module power-source supply.
	VDD	8	-	Power	Positive supply for I/O and some logic.
	VDD	20	-	Power	Positive supply for I/O and some logic.
	VDD	32	-	Power	Positive supply for I/O and some logic.
	VDD	44	-	Power	Positive supply for I/O and some logic.
	VDD	56	-	Power	Positive supply for I/O and some logic.
	VDD	68	-	Power	Positive supply for I/O and some logic.
	VDD	81	-	Power	Positive supply for I/O and some logic.
	VDD	93	-	Power	Positive supply for I/O and some logic.
VDD25	14	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.	
VDD25	38	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.	
VDD25	62	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.	

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
	VDD25	88	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
	VDDA	3	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
	VDDA	98	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
	$\overline{\text{WAKE}}$	50	I	OD	An external input that brings the processor out of hibernate mode when asserted.
SSI	SSI0Clk	28	I/O	TTL	SSI module 0 clock
	SSI0Fss	29	I/O	TTL	SSI module 0 frame
	SSI0Rx	30	I	TTL	SSI module 0 receive
	SSI0Tx	31	O	TTL	SSI module 0 transmit
System Control & Clocks	CMOD0	65	I/O	TTL	CPU Mode bit 0. Input must be set to logic 0 (grounded); other encodings reserved.
	CMOD1	76	I/O	TTL	CPU Mode bit 1. Input must be set to logic 0 (grounded); other encodings reserved.
	OSC0	48	I	Analog	Main oscillator crystal input or an external clock reference input.
	OSC1	49	O	Analog	Main oscillator crystal output.
	$\overline{\text{RST}}$	64	I	TTL	System reset input.
	$\overline{\text{TRST}}$	89	I	TTL	JTAG TRSTn
	XOSC0	52	I	Analog	Hibernation Module oscillator crystal input or an external clock reference input. Note that this is either a 4.19-MHz crystal or a 32.768-kHz oscillator for the Hibernation Module RTC. See the CLKSEL bit in the HIBCTL register.
UART	XOSC1	53	O	Analog	Hibernation Module oscillator crystal output.
	U0Rx	26	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
	U0Tx	27	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
	U1Rx	12	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U1Tx	13	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	U2Rx	19	I	TTL	UART 2 Receive. When in IrDA mode, this signal has IrDA modulation.
U2Tx	18	O	TTL	UART 2 Transmit. When in IrDA mode, this signal has IrDA modulation.	

Table 20-4. GPIO Pins and Alternate Functions

GPIO Pin	Pin Number	Multiplexed Function	Multiplexed Function
PA0	26	U0Rx	

GPIO Pin	Pin Number	Multiplexed Function	Multiplexed Function
PA1	27	U0Tx	
PA2	28	SSI0Clk	
PA3	29	SSI0Fss	
PA4	30	SSI0Rx	
PA5	31	SSI0Tx	
PA6	34	I2C1SCL	
PA7	35	I2C1SDA	
PB0	66	CCP0	
PB1	67	CCP2	
PB2	70	I2C0SCL	
PB3	71	I2C0SDA	
PB4	92	C0-	
PB5	91	C1-	
PB6	90	C0+	C0o
PB7	89	TRST	
PC0	80	TCK	SWCLK
PC1	79	TMS	SWDIO
PC2	78	TDI	
PC3	77	TDO	SWO
PC4	25	CCP5	
PC5	24	C1+	C1o
PC6	23	C2+	C2o
PC7	22	C2-	
PD0	10	CAN0Rx	
PD1	11	CAN0Tx	
PD2	12	U1Rx	
PD3	13	U1Tx	
PE0	72	CCP3	
PE1	73		
PE2	74	CCP4	
PE3	75	CCP1	
PF0	47		
PF1	61		
PF2	60	LED1	
PF3	59	LED0	
PG0	19	U2Rx	
PG1	18	U2Tx	

21 Operating Characteristics

Table 21-1. Temperature Characteristics

Characteristic	Symbol	Value	Unit
Operating temperature range ^a	T_A	-40 to +85	°C

a. Maximum storage temperature is 150°C.

Table 21-2. Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance (junction to ambient) ^a	Θ_{JA}	55.3	°C/W
Average junction temperature ^b	T_J	$T_A + (P_{AVG} \cdot \Theta_{JA})$	°C

a. Junction to ambient thermal resistance Θ_{JA} numbers are determined by a package simulator.

b. Power dissipation is a function of temperature.

22 Electrical Characteristics

22.1 DC Characteristics

22.1.1 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 22-1. Maximum Ratings

Characteristic ^a	Symbol	Value		Unit
		Min	Max	
I/O supply voltage (V_{DD})	V_{DD}	0	4	V
Core supply voltage (V_{DD25})	V_{DD25}	0	4	V
Analog supply voltage (V_{DDA})	V_{DDA}	0	4	V
Battery supply voltage (V_{BAT})	V_{BAT}	0	4	V
Ethernet PHY supply voltage (V_{CCPHY})	V_{CCPHY}	0	4	V
Input voltage	V_{IN}	-0.3	5.5	V
Maximum current per output pins	I	-	25	mA

a. Voltages are measured with respect to GND.

Important: This device contains circuitry to protect the inputs against damage due to high-static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either GND or V_{DD}).

22.1.2 Recommended DC Operating Conditions

Table 22-2. Recommended DC Operating Conditions

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{DD}	I/O supply voltage	3.0	3.3	3.6	V
V_{DD25}	Core supply voltage	2.25	2.5	2.75	V
V_{DDA}	Analog supply voltage	3.0	3.3	3.6	V
V_{BAT}	Battery supply voltage	2.3	3.0	3.6	V
V_{CCPHY}	Ethernet PHY supply voltage	3.0	3.3	3.6	V
V_{IH}	High-level input voltage	2.0	-	5.0	V
V_{IL}	Low-level input voltage	-0.3	-	1.3	V
V_{SIH}	High-level input voltage for Schmitt trigger inputs	$0.8 * V_{DD}$	-	V_{DD}	V
V_{SIL}	Low-level input voltage for Schmitt trigger inputs	0	-	$0.2 * V_{DD}$	V
V_{OH}	High-level output voltage	2.4	-	-	V
V_{OL}	Low-level output voltage	-	-	0.4	V

Parameter	Parameter Name	Min	Nom	Max	Unit
I _{OH}	High-level source current, V _{OH} =2.4 V				
	2-mA Drive	2.0	-	-	mA
	4-mA Drive	4.0	-	-	mA
	8-mA Drive	8.0	-	-	mA
I _{OL}	Low-level sink current, V _{OL} =0.4 V				
	2-mA Drive	2.0	-	-	mA
	4-mA Drive	4.0	-	-	mA
	8-mA Drive	8.0	-	-	mA

22.1.3 On-Chip Low Drop-Out (LDO) Regulator Characteristics

Table 22-3. LDO Regulator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
V _{LDOOUT}	Programmable internal (logic) power supply output value	2.25	2.5	2.75	V
	Output voltage accuracy	-	2%	-	%
t _{PON}	Power-on time	-	-	100	μs
t _{ON}	Time on	-	-	200	μs
t _{OFF}	Time off	-	-	100	μs
V _{STEP}	Step programming incremental voltage	-	50	-	mV
C _{LDO}	External filter capacitor size for internal power supply	-	1	-	μF

22.1.4 Power Specifications

The power measurements specified in the tables that follow are run on the core processor using SRAM with the following specifications (except as noted):

- V_{DD} = 3.3 V
- V_{DD25} = 2.50 V
- V_{BAT} = 3.0 V
- V_{DDA} = 3.3 V
- V_{DDPHY} = 3.3 V
- Temperature = 25°C
- Clock Source (MOSC) = 3.579545 MHz Crystal Oscillator
- Main oscillator (MOSC) = enabled
- Internal oscillator (IOSC) = disabled

22.1.5 Flash Memory Characteristics

Table 22-4. Flash Memory Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
PE _{CYC}	Number of guaranteed program/erase cycles before failure ^a	10,000	100,000	-	cycles
T _{RET}	Data retention at average operating temperature of 85°C	10	-	-	years
T _{PROG}	Word program time	20	-	-	µs
T _{ERASE}	Page erase time	20	-	-	ms
T _{ME}	Mass erase time	200	-	-	ms

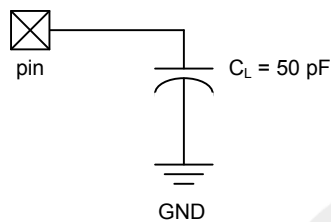
a. A program/erase cycle is defined as switching the bits from 1-> 0 -> 1.

22.2 AC Characteristics

22.2.1 Load Conditions

Unless otherwise specified, the following conditions are true for all timing measurements. Timing measurements are for 4-mA drive strength.

Figure 22-1. Load Conditions



22.2.2 Clocks

Table 22-5. Phase Locked Loop (PLL) Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
f _{ref_crystal}	Crystal reference ^a	3.579545	-	8.192	MHz
f _{ref_ext}	External clock reference ^a	3.579545	-	8.192	MHz
f _{pll}	PLL frequency ^b	-	400	-	MHz
T _{READY}	PLL lock time	-	-	0.5	ms

a. The exact value is determined by the crystal value programmed into the XTAL field of the **Run-Mode Clock Configuration (RCC)** register.

b. PLL frequency is automatically calculated by the hardware based on the XTAL field of the **RCC** register.

Table 22-6. Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
f _{IOSC}	Internal 12 MHz oscillator frequency	8.4	12	15.6	MHz
f _{IOSC30KHZ}	Internal 30 KHz oscillator frequency	21	30	39	KHz
f _{XOSC}	Hibernation module oscillator frequency	-	4.194304	-	MHz
f _{XOSC_XTAL}	Crystal reference for hibernation oscillator	-	4.194304	-	MHz
f _{XOSC_EXT}	External clock reference for hibernation module	-	32.768	-	KHz

Parameter	Parameter Name	Min	Nom	Max	Unit
f _{MOSC}	Main oscillator frequency	1	-	8	MHz
t _{MOSC_per}	Main oscillator period	125	-	1000	ns
f _{ref_crystal_bypass}	Crystal reference using the main oscillator (PLL in BYPASS mode) ^a	1	-	8	MHz
f _{ref_ext_bypass}	External clock reference (PLL in BYPASS mode) ^a	0	-	50	MHz
f _{system_clock}	System clock	0	-	50	MHz

a. The ADC must be clocked from the PLL or directly from a 14-MHz to 18-MHz clock source to operate properly.

Table 22-7. Crystal Characteristics

Parameter Name	Value				Units
	8	6	4	3.5	
Frequency	8	6	4	3.5	MHz
Frequency tolerance	±50	±50	±50	±50	ppm
Aging	±5	±5	±5	±5	ppm/yr
Oscillation mode	Parallel	Parallel	Parallel	Parallel	
Temperature stability (0 - 85 °C)	±25	±25	±25	±25	ppm
Motional capacitance (typ)	27.8	37.0	55.6	63.5	pF
Motional inductance (typ)	14.3	19.1	28.6	32.7	mH
Equivalent series resistance (max)	120	160	200	220	Ω
Shunt capacitance (max)	10	10	10	10	pF
Load capacitance (typ)	16	16	16	16	pF
Drive level (typ)	100	100	100	100	μW

22.2.3 Temperature Sensor

Table 22-8. Temperature Sensor Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
V _{TSO}	Output voltage	0.3	-	2.7	V
t _{TSERR}	Output voltage temperature accuracy	-	-	±3.5	°C
t _{TSNL}	Output temperature nonlinearity	-	-	±1	°C

22.2.4 Analog-to-Digital Converter

Table 22-9. ADC Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
V _{ADCIN}	Maximum single-ended, full-scale analog input voltage	-	-	3.0	V
	Minimum single-ended, full-scale analog input voltage	-	-	0	V
	Maximum differential, full-scale analog input voltage	-	-	1.5	V
	Minimum differential, full-scale analog input voltage	-	-	-1.5	V
C _{ADCIN}	Equivalent input capacitance	-	1	-	pF
N	Resolution	-	10	-	bits
f _{ADC}	ADC internal clock frequency	14	16	18	MHz
t _{ADCCONV}	Conversion time	-	-	16	t _{ADCcycles} ^a
f _{ADCCONV}	Conversion rate	875	1000	1125	k samples/s

Parameter	Parameter Name	Min	Nom	Max	Unit
INL	Integral nonlinearity	-	-	±1	LSB
DNL	Differential nonlinearity	-	-	±1	LSB
OFF	Offset	-	-	±1	LSB
GAIN	Gain	-	-	±1	LSB

a. $t_{ADC} = 1/f_{ADC \text{ clock}}$

22.2.5 Analog Comparator

Table 22-10. 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 22-11. Analog Comparator Voltage Reference Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
R_{HR}	Resolution high range	-	$V_{DD}/32$	-	LSB
R_{LR}	Resolution low range	-	$V_{DD}/24$	-	LSB
A_{HR}	Absolute accuracy high range	-	-	±1/2	LSB
A_{LR}	Absolute accuracy low range	-	-	±1/4	LSB

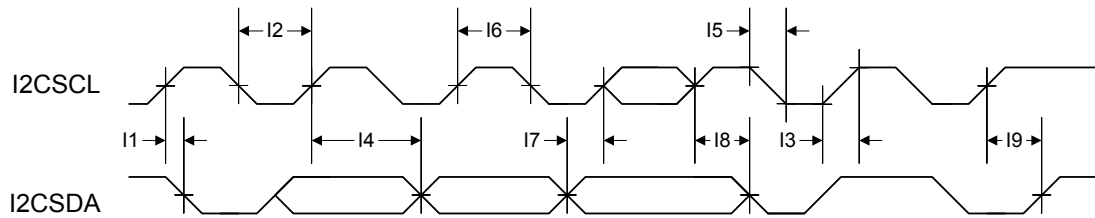
22.2.6 I²C

Table 22-12. I²C Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
11 ^a	t_{SCH}	Start condition hold time	36	-	-	system clocks
12 ^a	t_{LP}	Clock Low period	36	-	-	system clocks
13 ^b	t_{SRT}	I ² C SCL/I ² C SDA rise time ($V_{IL}=0.5$ V to $V_{IH}=2.4$ V)	-	-	(see note b)	ns
14 ^a	t_{DH}	Data hold time	2	-	-	system clocks
15 ^c	t_{SFT}	I ² C SCL/I ² C SDA fall time ($V_{IH}=2.4$ V to $V_{IL}=0.5$ V)	-	9	10	ns
16 ^a	t_{HT}	Clock High time	24	-	-	system clocks
17 ^a	t_{DS}	Data setup time	18	-	-	system clocks
18 ^a	t_{SCSR}	Start condition setup time (for repeated start condition only)	36	-	-	system clocks
19 ^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 (I2CMTPR) register; a TPR programmed for the maximum I²C SCL 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 I²C SCL 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 I²C SCL and I²C SDA are open-drain-type outputs, which the controller can only actively drive Low, the time I²C SCL or I²C SDA 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 22-2. I²C Timing



22.2.7 Ethernet Controller

Table 22-13. 100BASE-TX Transmitter Characteristics^a

Parameter Name	Min	Nom	Max	Unit
Peak output amplitude	950	-	1050	mVpk
Output amplitude symmetry	0.98	-	1.02	mVpk
Output overshoot	-	-	5	%
Rise/Fall time	3	-	5	ns
Rise/Fall time imbalance	-	-	500	ps
Duty cycle distortion	-	-	-	ps
Jitter	-	-	1.4	ns

a. Measured at the line side of the transformer.

Table 22-14. 100BASE-TX Transmitter Characteristics (informative)^a

Parameter Name	Min	Nom	Max	Unit
Return loss	16	-	-	dB
Open-circuit inductance	350	-	-	μs

a. The specifications in this table are included for information only. They are mainly a function of the external transformer and termination resistors used for measurements.

Table 22-15. 100BASE-TX Receiver Characteristics

Parameter Name	Min	Nom	Max	Unit
Signal detect assertion threshold	600	700	-	mVppd
Signal detect de-assertion threshold	350	425	-	mVppd
Differential input resistance	20	-	-	kΩ
Jitter tolerance (pk-pk)	4	-	-	ns
Baseline wander tracking	-75	-	+75	%
Signal detect assertion time	-	-	1000	μs
Signal detect de-assertion time	-	-	4	μs

Table 22-16. 10BASE-T Transmitter Characteristics^a

Parameter Name	Min	Nom	Max	Unit
Peak differential output signal	2.2	-	2.8	V
Harmonic content	27	-	-	dB
Link pulse width	-	100	-	ns

Parameter Name	Min	Nom	Max	Unit
Start-of-idle pulse width	-	300	-	ns
		350		

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 22-17. 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 22-18. 10BASE-T Receiver Characteristics

Parameter Name	Min	Nom	Max	Unit
DLL phase acquisition time	-	10	-	BT
Jitter tolerance (pk-pk)	30	-	-	ns
Input squelched threshold	500	600	700	mVppd
Input unsquelched threshold	275	350	425	mVppd
Differential input resistance	-	20	-	k Ω
Bit error ratio	-	10 ⁻¹⁰	-	-
Common-mode rejection	25	-	-	V

Table 22-19. Isolation Transformers^a

Name	Value	Condition
Turns ratio	1 CT : 1 CT	+/- 5%
Open-circuit inductance	350 μ H (min)	@ 10 mV, 10 kHz
Leakage inductance	0.40 μ H (max)	@ 1 MHz (min)
Inter-winding capacitance	25 pF (max)	
DC resistance	0.9 Ohm (max)	
Insertion loss	0.4 dB (typ)	0-65 MHz
HIPOT	1500	Vrms

a. Two simple 1:1 isolation transformers are required at the line interface. Transformers with integrated common-mode chokes are recommended for exceeding FCC requirements. This table gives the recommended line transformer characteristics.

Note: The 100Base-TX amplitude specifications assume a transformer loss of 0.4 dB. For the transmit line transformer with higher insertion losses, up to 1.2 dB of insertion loss can be compensated by selecting the appropriate setting in the Transmit Amplitude Selection (TXO) bits in the **MR19** register.

Table 22-20. Ethernet Reference Crystal^a

Name	Value	Condition
Frequency	25.00000	MHz
Load capacitance ^b	4 ^c	pF
Frequency tolerance	±50	PPM
Aging	±2	PPM/yr
Temperature stability (0° to 70°)	±5	PPM
Oscillation mode	Parallel resonance, fundamental mode	
Parameters at 25° C ±2° C; Drive level = 0.5 mW		
Drive level (typ)	50-100	µW
Shunt capacitance (max)	10	pF
Motional capacitance (min)	10	fF
Serious resistance (max)	60	Ω
Spurious response (max)	> 5 dB below main within 500 kHz	

- a. If the internal crystal oscillator is used, select a crystal with the following characteristics.
- b. Equivalent differential capacitance across XTLP/XTLN.
- c. If crystal with a larger load is used, external shunt capacitors to ground should be added to make up the equivalent capacitance difference.

Figure 22-3. External XTLP Oscillator Characteristics

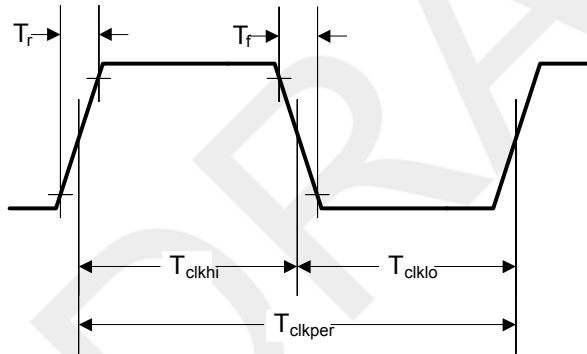


Table 22-21. External XTLP Oscillator Characteristics

Parameter Name	Symbol	Min	Nom	Max	Unit
XTLN Input Low Voltage	$XTLN_{ILV}$	-	-	0.8	-
XTLP Frequency ^a	$XTLP_f$	-	25.0	-	-
XTLP Period ^b	T_{clkper}	-	40	-	-
XTLP Duty Cycle	$XTLP_{DC}$	40	-	60	%
		40	-	60	
Rise/Fall Time	T_r, T_f	-	-	4.0	ns
Absolute Jitter		-	-	0.1	ns

a. IEEE 802.3 frequency tolerance ±50 ppm.

b. IEEE 802.3 frequency tolerance ± 50 ppm.

22.2.8 Hibernation Module

The Hibernation Module requires special system implementation considerations since it is intended to power-down all other sections of its host device. The system power-supply distribution and interfaces of the system must be driven to $0 V_{DC}$ or powered down with the same regulator controlled by \overline{HIB} .

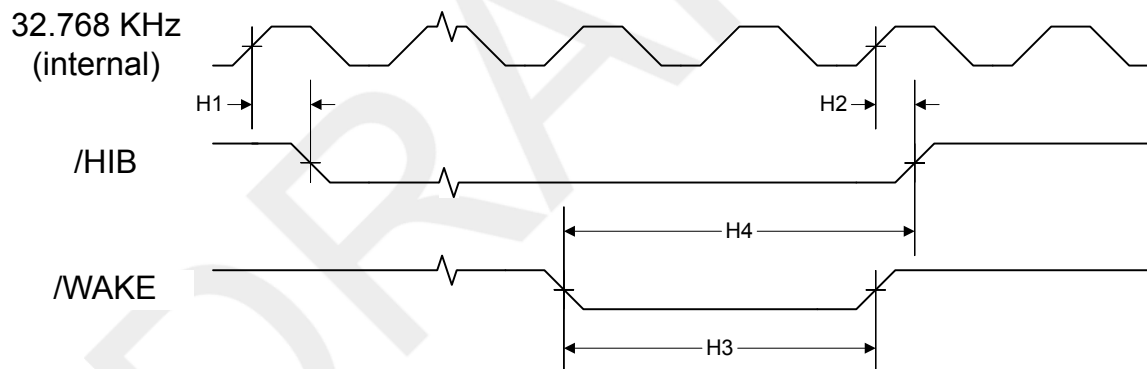
The regulators controlled by \overline{HIB} are expected to have a settling time of 250 μs or less.

Table 22-22. Hibernation Module Characteristics

Parameter No	Parameter	Parameter Name	Min	Nom	Max	Unit
H1	t_{HIB_LOW}	Internal 32.768 KHz clock reference rising edge to \overline{HIB} asserted	-	200	-	μs
H2	t_{HIB_HIGH}	Internal 32.768 KHz clock reference rising edge to \overline{HIB} deasserted	-	30	-	μs
H3	t_{WAKE_ASSERT}	\overline{WAKE} assertion time	62	-	-	μs
H4	$t_{WAKETOHIB}$	\overline{WAKE} assert to \overline{HIB} desassert	62	-	124	μs
H5	t_{XOSC_SETTLE}	XOSC settling time ^a	20	-	-	ms
H6	$t_{HIB_REG_WRITE}$	Time for a write to non-volatile registers in HIB module to complete	92	-	-	μs

a. This parameter is highly sensitive to PCB layout and trace lengths, which may make this parameter time longer. Care must be taken in PCB design to minimize trace lengths and RLC (resistance, inductance, capacitance).

Figure 22-4. Hibernation Module Timing



22.2.9 Synchronous Serial Interface (SSI)

Table 22-23. SSI Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
S1	t_{clk_per}	SSIClk cycle time	2	-	65024	system clocks
S2	t_{clk_high}	SSIClk high time	-	1/2	-	t_{clk_per}
S3	t_{clk_low}	SSIClk low time	-	1/2	-	t_{clk_per}
S4	t_{clkrf}	SSIClk rise/fall time	-	7.4	26	ns
S5	t_{DMd}	Data from master valid delay time	0	-	20	ns
S6	t_{DMs}	Data from master setup time	20	-	-	ns
S7	t_{DMh}	Data from master hold time	40	-	-	ns
S8	t_{DSs}	Data from slave setup time	20	-	-	ns

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
S9	t_{DSh}	Data from slave hold time	40	-	-	ns

Figure 22-5. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement

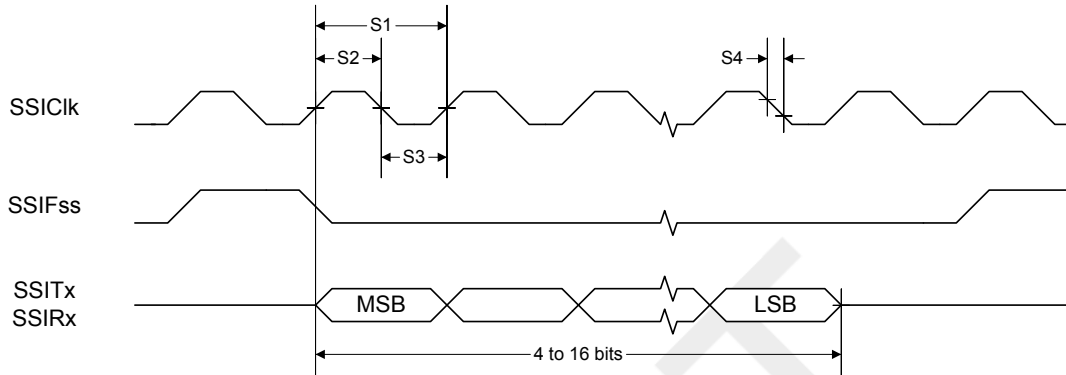


Figure 22-6. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer

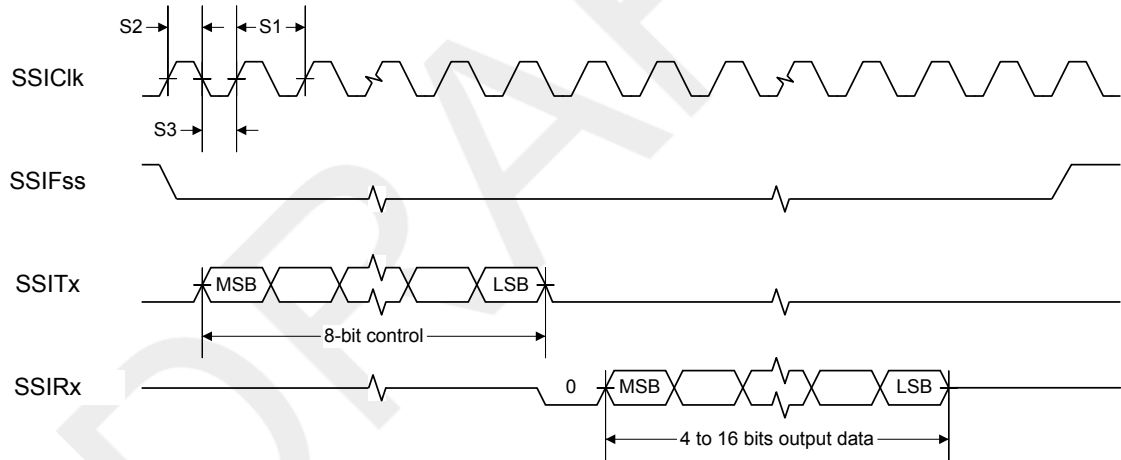
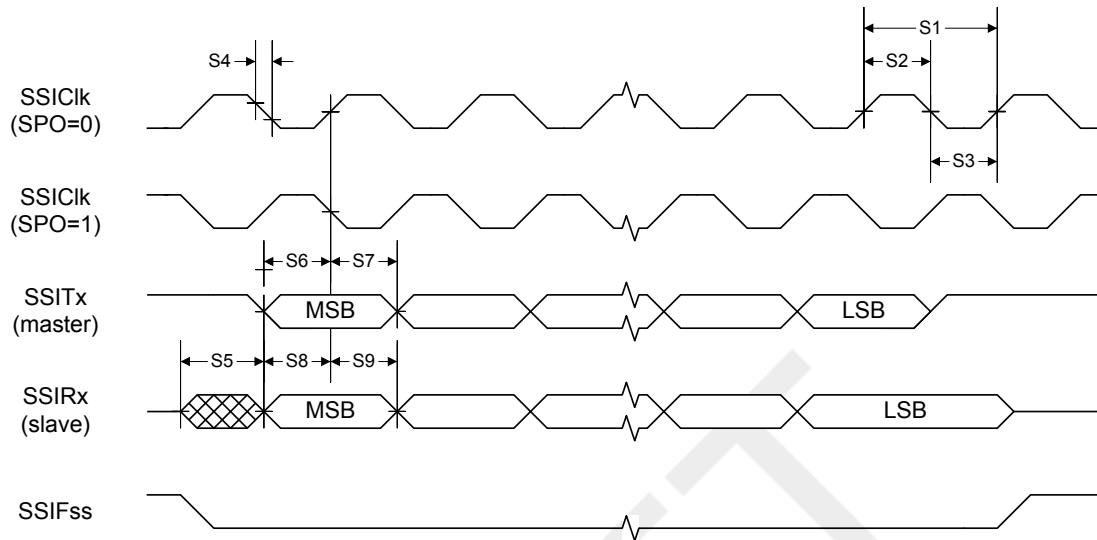


Figure 22-7. SSI Timing for SPI Frame Format (FRF=00), with SPH=1



22.2.10 JTAG and Boundary Scan

Table 22-24. JTAG Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J1	f_{TCK}	TCK operational clock frequency	0	-	10	MHz
J2	t_{TCK}	TCK operational clock period	100	-	-	ns
J3	t_{TCK_LOW}	TCK clock Low time	-	t_{TCK}	-	ns
J4	t_{TCK_HIGH}	TCK clock High time	-	t_{TCK}	-	ns
J5	t_{TCK_R}	TCK rise time	0	-	10	ns
J6	t_{TCK_F}	TCK fall time	0	-	10	ns
J7	t_{TMS_SU}	TMS setup time to TCK rise	20	-	-	ns
J8	t_{TMS_HLD}	TMS hold time from TCK rise	20	-	-	ns
J9	t_{TDI_SU}	TDI setup time to TCK rise	25	-	-	ns
J10	t_{TDI_HLD}	TDI hold time from TCK rise	25	-	-	ns
J11	t_{TDO_ZDV}	TCK fall to Data Valid from High-Z	-	23	35	ns
		2-mA drive		15	26	ns
		4-mA drive		14	25	ns
		8-mA drive with slew rate control		18	29	ns
J12	t_{TDO_DV}	TCK fall to Data Valid from Data Valid	-	21	35	ns
		2-mA drive		14	25	ns
		4-mA drive		13	24	ns
		8-mA drive with slew rate control		18	28	ns

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J13	TCK fall to High-Z from Data Valid	2-mA drive	-	9	11	ns
		4-mA drive		7	9	ns
		8-mA drive		6	8	ns
		8-mA drive with slew rate control		7	9	ns
J14	t_{TRST}	\overline{TRST} assertion time	100	-	-	ns
J15	t_{TRST_SU}	\overline{TRST} setup time to TCK rise	10	-	-	ns

Figure 22-8. JTAG Test Clock Input Timing

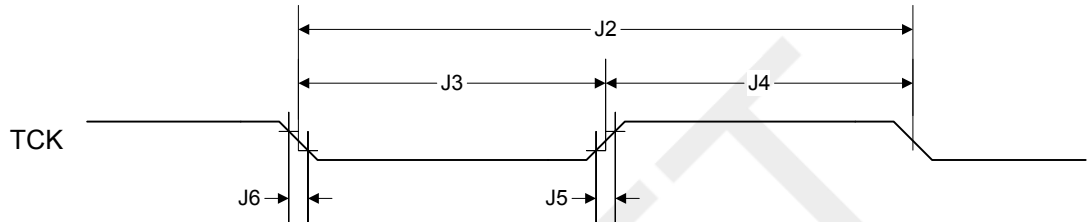


Figure 22-9. JTAG Test Access Port (TAP) Timing

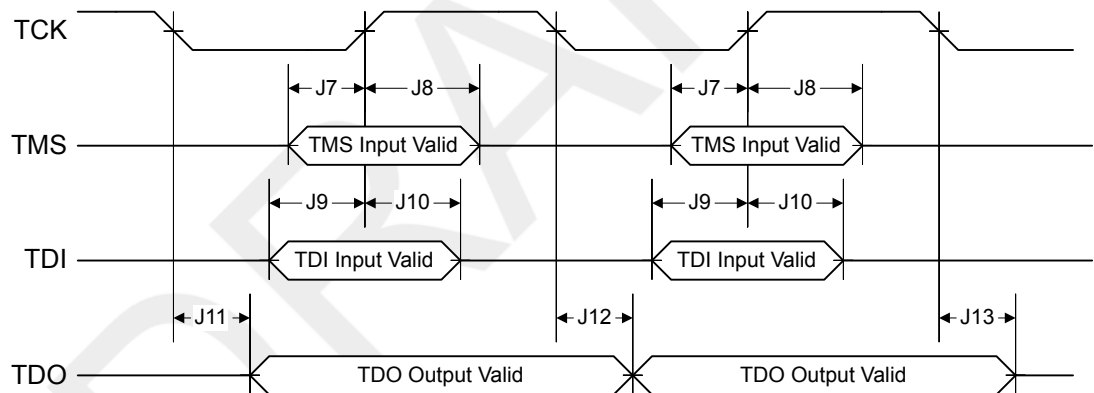
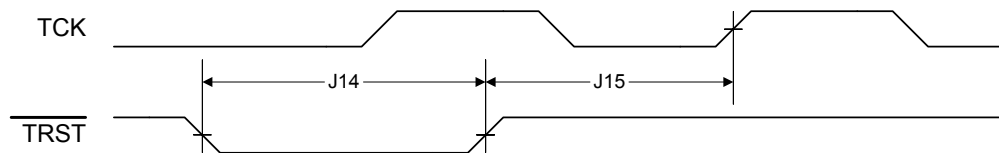


Figure 22-10. JTAG TRST Timing



22.2.11 General-Purpose I/O

Note: All GPIOs are 5 V-tolerant.

Table 22-25. GPIO Characteristics

Parameter	Parameter Name	Condition	Min	Nom	Max	Unit
$t_{\text{GPIO R}}$	GPIO Rise Time (from 20% to 80% of V_{DD})	2-mA drive	-	17	26	ns
		4-mA drive		9	13	ns
		8-mA drive		6	9	ns
		8-mA drive with slew rate control		10	12	ns
$t_{\text{GPIO F}}$	GPIO Fall Time (from 80% to 20% of V_{DD})	2-mA drive	-	17	25	ns
		4-mA drive		8	12	ns
		8-mA drive		6	10	ns
		8-mA drive with slew rate control		11	13	ns

22.2.12 Reset

Table 22-26. Reset Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
R1	V_{TH}	Reset threshold	-	2.0	-	V
R2	V_{BTH}	Brown-Out threshold	2.85	2.9	2.95	V
R3	T_{POR}	Power-On Reset timeout	-	10	-	ms
R4	T_{BOR}	Brown-Out timeout	-	500	-	μs
R5	T_{IRPOR}	Internal reset timeout after POR	6	-	11	ms
R6	T_{IRBOR}	Internal reset timeout after BOR ^a	0	-	1	μs
R7	T_{IRHWR}	Internal reset timeout after hardware reset ($\overline{\text{RST}}$ pin)	0	-	1	ms
R8	T_{IRSWR}	Internal reset timeout after software-initiated system reset ^a	2.5	-	20	μs
R9	T_{IRWDR}	Internal reset timeout after watchdog reset ^a	2.5	-	20	μs
R10	T_{VDDRISE}	Supply voltage (V_{DD}) rise time (0V-3.3V)	-	-	100	ms
R11	T_{MIN}	Minimum $\overline{\text{RST}}$ pulse width	2	-	-	μs

a. $20 * t_{\text{MOSC_per}}$

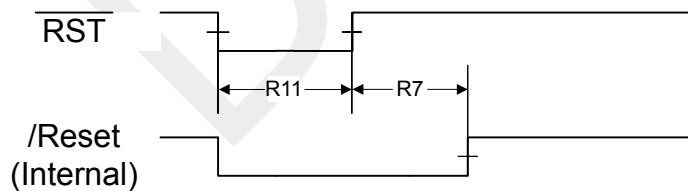
Figure 22-11. External Reset Timing ($\overline{\text{RST}}$)

Figure 22-12. Power-On Reset Timing



Figure 22-13. Brown-Out Reset Timing

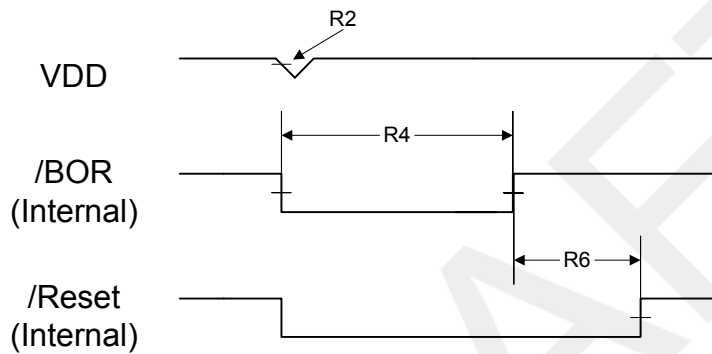


Figure 22-14. Software Reset Timing

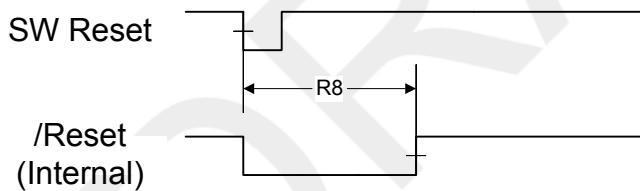
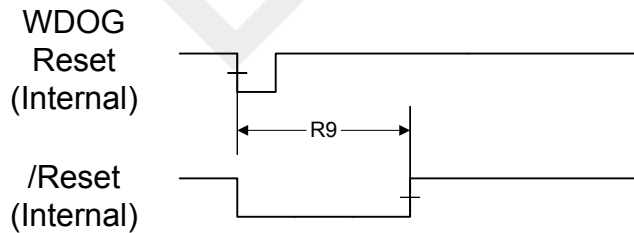
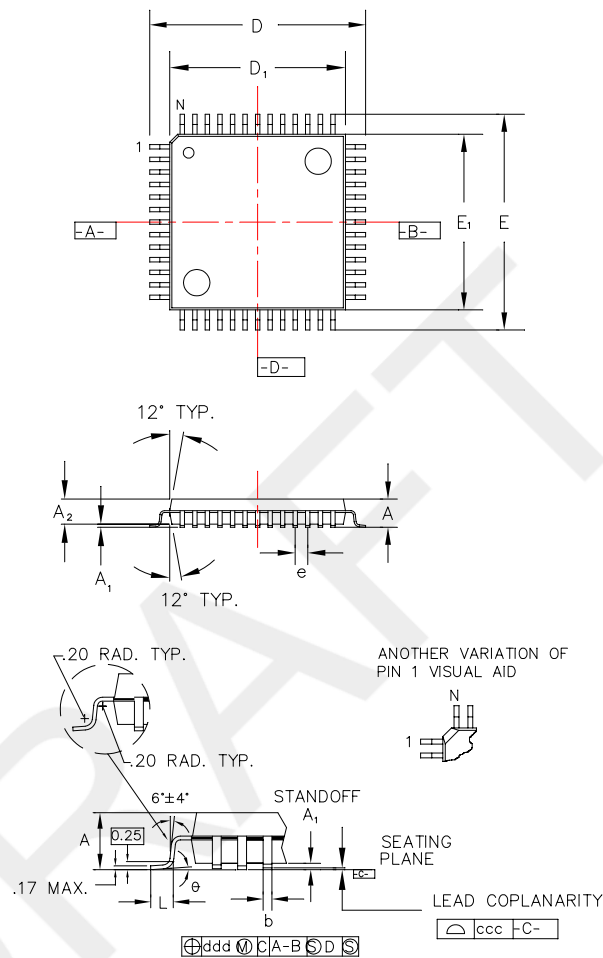


Figure 22-15. Watchdog Reset Timing



23 Package Information

Figure 23-1. 100-Pin LQFP Package



Notes

1. All dimensions shown in mm.
2. Dimensions shown are nominal with tolerances indicated.
3. Foot length 'L' is measured at gage plane 0.25 mm above seating plane.
4. L/F: Eftec 64T Cu or equivalent, 0.127 mm (0.005") or 0.152 mm (0.006") thick.
5. Use variation BED for body dimensions.

Body +2.00 mm Footprint, 1.4 mm package thickness		
Symbols	Leads	100L
A	Max.	1.60
A ₁		0.05 Min./0.15 Max.

A ₂	±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

DRAFT

24 Ordering and Contact Information

24.1 Ordering Information

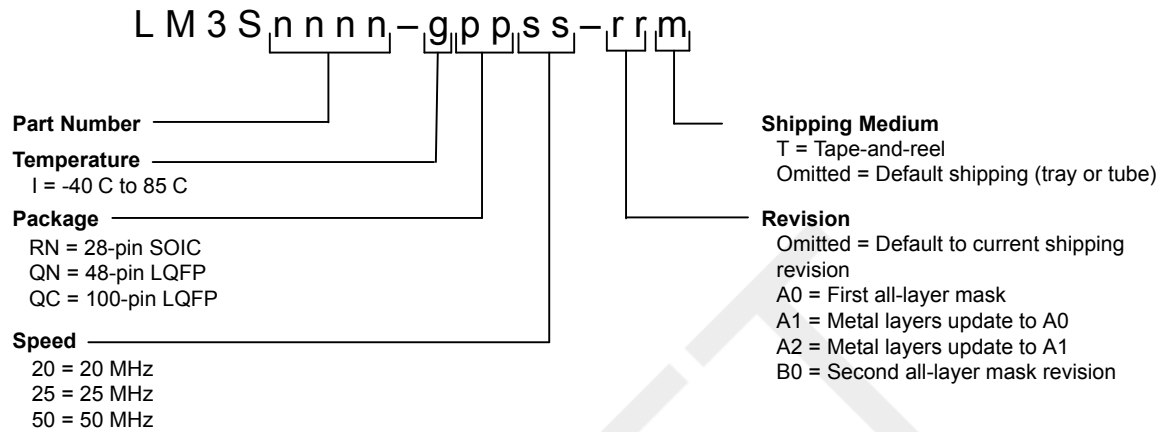


Table 24-1. Part Ordering Information

Orderable Part Number	Description
LM3S8938-IQC50	Stellaris® LM3S8938 Microcontroller

24.2 Company Information

Luminary Micro, Inc. designs, markets, and sells ARM Cortex-M3-based microcontrollers (MCUs). Austin, Texas-based Luminary Micro is the lead partner for the Cortex-M3 processor, delivering the world's first silicon implementation of the Cortex-M3 processor. Luminary Micro's introduction of the Stellaris® family of products provides 32-bit performance for the same price as current 8- and 16-bit microcontroller designs. With entry-level pricing at \$1.00 for an ARM technology-based MCU, Luminary Micro's Stellaris product line allows for standardization that eliminates future architectural upgrades or software tool changes.

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

24.3 Support Information

For support on Luminary Micro products, contact:
support@luminarymicro.com +1-512-279-8800, ext. 3

A Serial Flash Loader

A.1 Serial Flash Loader

The Stellaris[®] serial flash loader is a preprogrammed flash-resident utility used to download code to the flash memory of a device without the use of a debug interface. The serial flash loader uses a simple packet interface to provide synchronous communication with the device. The flash loader runs off the crystal and does not enable the PLL, so its speed is determined by the crystal used. The two serial interfaces that can be used are the UART0 and SSI interfaces. For simplicity, both the data format and communication protocol are identical for both serial interfaces.

A.2 Interfaces

Once communication with the flash loader is established via one of the serial interfaces, that interface is used until the flash loader is reset or new code takes over. For example, once you start communicating using the SSI port, communications with the flash loader via the UART are disabled until the device is reset.

A.2.1 UART

The Universal Asynchronous Receivers/Transmitters (UART) communication uses a fixed serial format of 8 bits of data, no parity, and 1 stop bit. The baud rate used for communication is automatically detected by the flash loader and can be any valid baud rate supported by the host and the device. The auto detection sequence requires that the baud rate should be no more than 1/32 the crystal frequency of the board that is running the serial flash loader. This is actually the same as the hardware limitation for the maximum baud rate for any UART on a Stellaris[®] device.

In order to determine the baud rate, the serial flash loader needs to determine the relationship between its own crystal frequency and the baud rate. This is enough information for the flash loader to configure its UART to the same baud rate as the host. This automatic baud-rate detection allows the host to use any valid baud rate that it wants to communicate with the device.

The method used to perform this automatic synchronization relies on the host sending the flash loader two bytes that are both 0x55. This generates a series of pulses to the flash loader that it can use to calculate the ratios needed to program the UART to match the host's baud rate. After the host sends the pattern, it attempts to read back one byte of data from the UART. The flash loader returns the value of 0xCC to indicate successful detection of the baud rate. If this byte is not received after at least twice the time required to transfer the two bytes, the host can resend another pattern of 0x55, 0x55, and wait for the 0xCC byte again until the flash loader acknowledges that it has received a synchronization pattern correctly. For example, the time to wait for data back from the flash loader should be calculated as at least $2 * (20(\text{bits/sync}) / \text{baud rate} (\text{bits/sec}))$. For a baud rate of 115200, this time is $2 * (20 / 115200)$ or 0.35 ms.

A.2.2 SSI

The Synchronous Serial Interface (SSI) port also uses a fixed serial format for communications, with the framing defined as Motorola format with SPH set to 1 and SPO set to 1. See the section on SSI formats for more details on this transfer protocol. Like the UART, this interface has hardware requirements that limit the maximum speed that the SSI clock can run. This allows the SSI clock to be at most 1/12 the crystal frequency of the board running the flash loader. Since the host device is the master, the SSI on the flash loader device does not need to determine the clock as it is provided directly by the host.

A.3 Packet Handling

All communications, with the exception of the UART auto-baud, are done via defined packets that are acknowledged (ACK) or not acknowledged (NAK) by the devices. The packets use the same format for receiving and sending packets, including the method used to acknowledge successful or unsuccessful reception of a packet.

A.3.1 Packet Format

All packets sent and received from the device use the following byte-packed format.

```
struct
{
  unsigned char ucSize;
  unsigned char ucChecksum;
  unsigned char Data[];
};
```

ucSize	The first byte received holds the total size of the transfer including the size and checksum bytes.
ucChecksum	This holds a simple checksum of the bytes in the data buffer only. The algorithm is $Data[0]+Data[1]+\dots+Data[ucSize-3]$.
Data	This is the raw data intended for the device, which is formatted in some form of command interface. There should be $ucSize-2$ bytes of data provided in this buffer to or from the device.

A.3.2 Sending Packets

The actual bytes of the packet can be sent individually or all at once; the only limitation is that commands that cause flash memory access should limit the download sizes to prevent losing bytes during flash programming. This limitation is discussed further in the commands that interact with the flash.

Once the packet has been formatted correctly by the host, it should be sent out over the UART or SSI interface. Then the host should poll the UART or SSI interface for the first non-zero data returned from the device. The first non-zero byte will either be an ACK (0xCC) or a NAK (0x33) byte from the device indicating the packet was received successfully (ACK) or unsuccessfully (NAK). This does not indicate that the actual contents of the command issued in the data portion of the packet were valid, just that the packet was received correctly.

A.3.3 Receiving Packets

The flash loader sends a packet of data in the same format that it receives a packet. The flash loader may transfer leading zero data before the first actual byte of data is sent out. The first non-zero byte is the size of the packet followed by a checksum byte, and finally followed by the data itself. There is no break in the data after the first non-zero byte is sent from the flash loader. Once the device communicating with the flash loader receives all the bytes, it must either ACK or NAK the packet to indicate that the transmission was successful. The appropriate response after sending a NAK to the flash loader is to resend the command that failed and request the data again. If needed, the host may send leading zeros before sending down the ACK/NAK signal to the flash loader, as the flash loader only accepts the first non-zero data as a valid response. This zero padding is needed by the SSI interface in order to receive data to or from the flash loader.

A.4 Commands

The next section defines the list of commands that can be sent to the flash loader. The first byte of the data should always be one of the defined commands, followed by data or parameters as determined by the command that is sent.

A.4.1 COMMAND_PING (0X20)

This command simply accepts the command and sets the global status to success. The format of the packet is as follows:

```
Byte[0] = 0x03;  
Byte[1] = checksum(Byte[2]);  
Byte[2] = COMMAND_PING;
```

The ping command has 3 bytes and the value for `COMMAND_PING` is 0x20 and the checksum of one byte is that same byte, making `Byte[1]` also 0x20. Since the ping command has no real return status, the receipt of an ACK can be interpreted as a successful ping to the flash loader.

A.4.2 COMMAND_GET_STATUS (0x23)

This command returns the status of the last command that was issued. Typically, this command should be sent after every command to ensure that the previous command was successful or to properly respond to a failure. The command requires one byte in the data of the packet and should be followed by reading a packet with one byte of data that contains a status code. The last step is to ACK or NAK the received data so the flash loader knows that the data has been read.

```
Byte[0] = 0x03  
Byte[1] = checksum(Byte[2])  
Byte[2] = COMMAND_GET_STATUS
```

A.4.3 COMMAND_DOWNLOAD (0x21)

This command is sent to the flash loader to indicate where to store data and how many bytes will be sent by the `COMMAND_SEND_DATA` commands that follow. The command consists of two 32-bit values that are both transferred MSB first. The first 32-bit value is the address to start programming data into, while the second is the 32-bit size of the data that will be sent. This command also triggers an erase of the full area to be programmed so this command takes longer than other commands. This results in a longer time to receive the ACK/NAK back from the board. This command should be followed by a `COMMAND_GET_STATUS` to ensure that the Program Address and Program size are valid for the device running the flash loader.

The format of the packet to send this command is a follows:

```
Byte[0] = 11  
Byte[1] = checksum(Bytes[2:10])  
Byte[2] = COMMAND_DOWNLOAD  
Byte[3] = Program Address [31:24]  
Byte[4] = Program Address [23:16]  
Byte[5] = Program Address [15:8]  
Byte[6] = Program Address [7:0]  
Byte[7] = Program Size [31:24]  
Byte[8] = Program Size [23:16]  
Byte[9] = Program Size [15:8]  
Byte[10] = Program Size [7:0]
```

A.4.4 COMMAND_SEND_DATA (0x24)

This command should only follow a `COMMAND_DOWNLOAD` command or another `COMMAND_SEND_DATA` command if more data is needed. Consecutive send data commands automatically increment address and continue programming from the previous location. The caller should limit transfers of data to a maximum 8 bytes of packet data to allow the flash to program successfully and not overflow input buffers of the serial interfaces. The command terminates programming once the number of bytes indicated by the `COMMAND_DOWNLOAD` command has been received. Each time this function is called it should be followed by a `COMMAND_GET_STATUS` to ensure that the data was successfully programmed into the flash. If the flash loader sends a NAK to this command, the flash loader does not increment the current address to allow retransmission of the previous data.

```
Byte[0] = 11
Byte[1] = checksum(Bytes[2:10])
Byte[2] = COMMAND_SEND_DATA
Byte[3] = Data[0]
Byte[4] = Data[1]
Byte[5] = Data[2]
Byte[6] = Data[3]
Byte[7] = Data[4]
Byte[8] = Data[5]
Byte[9] = Data[6]
Byte[10] = Data[7]
```

A.4.5 COMMAND_RUN (0x22)

This command is used to tell the flash loader to execute from the address passed as the parameter in this command. This command consists of a single 32-bit value that is interpreted as the address to execute. The 32-bit value is transmitted MSB first and the flash loader responds with an ACK signal back to the host device before actually executing the code at the given address. This allows the host to know that the command was received successfully and the code is now running.

```
Byte[0] = 7
Byte[1] = checksum(Bytes[2:6])
Byte[2] = COMMAND_RUN
Byte[3] = Execute Address[31:24]
Byte[4] = Execute Address[23:16]
Byte[5] = Execute Address[15:8]
Byte[6] = Execute Address[7:0]
```

A.4.6 COMMAND_RESET (0x25)

This command is used to tell the flash loader device to reset. This is useful when downloading a new image that overwrote the flash loader and wants to start from a full reset. Unlike the `COMMAND_RUN` command, this allows the initial stack pointer to be read by the hardware and set up for the new code. It can also be used to reset the flash loader if a critical error occurs and the host device wants to restart communication with the flash loader.

```
Byte[0] = 3
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_RESET
```

The flash loader responds with an ACK signal back to the host device before actually executing the software reset to the device running the flash loader. This allows the host to know that the command was received successfully and the part will be reset.

DRAFT

B Register Quick Reference

Name	Offset	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		
System Control																			
Base: 0x400F.E000																			
DID0 RO	0x000	VER						CLASS											
		MAJOR									MINOR								
PBORCTL R/W	0x030																		
		BORIOR																	
LDOPCTL R/W	0x034																		
		VADJ																	
RIS RO	0x050																		
		PLLLRIS BORRIS																	
IMC R/W	0x054																		
		PLLLIM BORIM																	
MISC R/W1C	0x058																		
		PLLLMIS BORMIS																	
RESC R/W	0x05C																		
		LDO SW WDT BOR POR EXT																	
RCC R/W	0x060																		
		ACG SYSDIV USESYDV																	
		PWRDN BYPASS XTAL OSCSRC IOSCDIS MOSCDIS																	
PLLCFG RO	0x064																		
		OD F R																	
RCC2 R/W	0x070																		
		USERCC2 SYSDIV2																	
		PWRDN2 BYPASS2 OSCSRC2																	
DSLPOLKCFG R/W	0x144																		
		DSDIVORIDE DSOSCSRC																	
DID1 RO	0x004																		
		VER						FAM						PARTNO					
		PINCOUNT						TEMP						PKG ROHS QUAL					
DC0 RO	0x008																		
		SRAMSZ																	
		FLASHSZ																	
DC1 RO	0x010																		
		SYSDIV						MAXADCSZ						CAN0 SARADC0					
		MPU HIB TEMPSNS PLL WDT SWO SWD JTAG																	
DC2 RO	0x014																		
		COMP2 COMP1 COMP0						TIMER3 TIMER2 TIMER1 TIMER0											
		I2C1 I2C0						SSI0 UART2 UART1 UART0											
DC3 RO	0x018																		
		CCP5 CCP4 CCP3 CCP2 CCP1 CCP0						ADC7 ADC6 ADC5 ADC4 ADC3 ADC2 ADC1 ADC0											
		C2O C2PLUS C2MINUS C1O C1PLUS C1MINUS C0O C0PLUS C0MINUS																	
DC4 RO	0x01C																		
		EPHY0 EMAC0						E1588											

Name	Offset	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							
RO											GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA							
RCGC0 R/W	0x100								CAN0								SARADC0							
											HIB			WDT										
SCGC0 R/W	0x110								CAN0								SARADC0							
											HIB			WDT										
DCGC0 R/W	0x120								CAN0								SARADC0							
											HIB			WDT										
RCGC1 R/W	0x104						COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0							
			I2C1		I2C0								SSI0		UART2	UART1	UART0							
SCGC1 R/W	0x114						COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0							
			I2C1		I2C0								SSI0		UART2	UART1	UART0							
DCGC1 R/W	0x124						COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0							
			I2C1		I2C0								SSI0		UART2	UART1	UART0							
RCGC2 R/W	0x108		EPHY0		EMAC0							GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA						
SCGC2 R/W	0x118		EPHY0		EMAC0							GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA						
DCGC2 R/W	0x128		EPHY0		EMAC0							GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA						
SRCR0 R/W	0x040								CAN0								SARADC0							
											HIB			WDT										
SRCR1 R/W	0x044						COMP2	COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0							
			I2C1		I2C0								SSI0		UART2	UART1	UART0							
SRCR2 R/W	0x048		EPHY0		EMAC0																			
												GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA						
Hibernation Module																								
HIBRTCC RO	0x000																RTCC							
																	RTCC							
HIBRTCM0 R/W	0x004																RTCM0							
																	RTCM0							
HIBRTCM1 R/W	0x008																RTCM1							
																	RTCM1							
HIBRTCLD R/W	0x00C																RTCLD							
																	RTCLD							
HIBCTL R/W	0x010																VABORT	CLK32EN	LOWBATTEN	PINWEN	RTCWEN	CLKSEL	HIBREQ	RTCEN
HIBIM	0x014																							

Name	Offset	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		
R/W														EXTW	LOWBAT	RTCAL1	RTCAL0		
HIBRIS RO	0x018													EXTW	LOWBAT	RTCAL1	RTCAL0		
HIBMIS RO	0x01C													EXTW	LOWBAT	RTCAL1	RTCAL0		
HIBIC W1C	0x020													EXTW	LOWBAT	RTCAL1	RTCAL0		
HIBRTCT R/W	0x024	TRIM																	
HIBDATA R/W	0x030-0x12C	RTD																	
Internal Memory																			
Base: 0x400F.D000																			
Base: 0x400F.E000																			
FMA R/W	0x000	OFFSET																	
FMD R/W	0x004	DATA																	
FMC R/W	0x008	WRKEY																	
FCRIS RO	0x00C													COMT	MERASE	ERASE	WRITE		
FCIM R/W	0x010															PMASK	AMASK		
FCMISC R/W1C	0x014															PMISC	AMISC		
USECRL R/W	0x140	USEC																	
FMPRE0 R/W	0x130 and 0x200	READ_ENABLE																	
FMPPE0 R/W	0x134 and 0x400	PROG_ENABLE																	
USER_DBG R/W	0x1D0	NOVWRTEN	DATA														INIT1	DBG1	DBG0
USER_REG0 R/W	0x1E0	NOVWRTEN	DATA																

Name	Offset	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
USER_REG1 R/W	0x1E4	NOWRITEN		DATA													
FMPRE1 R/W	0x204	DATA															
FMPRE2 R/W	0x208	READ_ENABLE															
FMPRE3 R/W	0x20C	READ_ENABLE															
FMPPE1 R/W	0x404	READ_ENABLE															
FMPPE2 R/W	0x408	READ_ENABLE															
FMPPE3 R/W	0x40C	READ_ENABLE															
FMPPE1 R/W	0x404	PROG_ENABLE															
FMPPE2 R/W	0x408	PROG_ENABLE															
FMPPE3 R/W	0x40C	PROG_ENABLE															
General-Purpose Input/Outputs (GPIOs)																	
Base: 0x4000.4000																	
Base: 0x4000.5000																	
Base: 0x4000.6000																	
Base: 0x4000.7000																	
Base: 0x4002.4000																	
Base: 0x4002.5000																	
Base: 0x4002.6000																	
GPIODATA R/W	0x000	DATA															
GPIODIR R/W	0x400	DIR															
GPIOIS R/W	0x404	IS															
GPIOIBE R/W	0x408	IBE															
GPIOIEV R/W	0x40C	IEV															
GPIOIM R/W	0x410	IME															
GPIORIS RO	0x414	RIS															
GPIOMIS RO	0x418	MIS															
GPIOICR	0x41C																

Name	Offset	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
W1C																	IC
GPIOAFSEL R/W	0x420																AFSEL
GPIODR2R R/W	0x500																DRV2
GPIODR4R R/W	0x504																DRV4
GPIODR8R R/W	0x508																DRV8
GPIODR R/W	0x50C																ODE
GPIOPUR R/W	0x510																PUE
GPIOPDR R/W	0x514																PDE
GPIOSLR R/W	0x518																SRL
GPIODEN R/W	0x51C																DEN
GPIOLOCK R/W	0x520								LOCK								LOCK
GPIOCR -	0x524																CR
GPIO_PID4 RO	0xFD0																PID4
GPIO_PID5 RO	0xFD4																PID5
GPIO_PID6 RO	0xFD8																PID6
GPIO_PID7 RO	0xFDC																PID7
GPIO_PID0 RO	0xFE0																PID0
GPIO_PID1 RO	0xFE4																PID1
GPIO_PID2 RO	0xFE8																PID2

Name	Offset	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	
GPIOBitID3 RO	0xFEC																	PID3
GPIOCellID0 RO	0xFF0																	CID0
GPIOCellID1 RO	0xFF4																	CID1
GPIOCellID2 RO	0xFF8																	CID2
GPIOCellID3 RO	0xFFC																	CID3
General-Purpose Timers																		
Base: 0x4003.0000																		
Base: 0x4003.1000																		
Base: 0x4003.2000																		
Base: 0x4003.3000																		
GPTMCFG R/W	0x000																	GPTMCFG
GPTMTAMR R/W	0x004														TAAMS	TACMR		TAMR
GPTMTBMR R/W	0x008														TBAMS	TBCMR		TBMR
GPTMCTL R/W	0x00C		TBPWML	TBOTE		TBEVENT	TBSTALL	TBEN		TAPWML	TAOTE	RTCEN		TAEVENT		TASTALL	TAEN	
GPTMIMR R/W	0x018						CBEIM	CBMIM	TBTOIM					RTCIM	CAEIM	CAMIM	TATOIM	
GPTMRIS RO	0x01C						CBERIS	CBMRIS	TBTORIS					RTCRIIS	CAERIS	CAMRIS	TATORIS	
GPTMMIS RO	0x020						CBEMIS	CBMMIS	TBTOMIS					RTCMIS	CAEMIS	CAMMIS	TATOMIS	
GPTMICR W1C	0x024						CBECINT	CBMCINT	TBTCOINT					RTCCOINT	CAECOINT	CAMCOINT	TATCOINT	
GPTMTALR R/W	0x028																	TAILRH TAILRL
GPTMTBILR R/W	0x02C																	TBILRL
GPTMACHR R/W	0x030																	TAMRH TAMRL
GPTMACHR	0x034																	

Name	Offset	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
R/W		TBMRL															
GPTMTAPR R/W	0x038	TAPSR															
GPTMTBPR R/W	0x03C	TBPSR															
GPTMAPMR R/W	0x040	TAPSMR															
GPTMBPMR R/W	0x044	TBPSMR															
GPTMTAR RO	0x048	TARH TARL															
GPTMTBR RO	0x04C	TBRL															
Watchdog Timer Base: 0x4000.0000																	
WDTLOAD R/W	0x000	WDTLoad															
WDTVALUE RO	0x004	WDTValue															
WDTCTL R/W	0x008	RESEN INTEN															
WDTICR WO	0x00C	WDTIntClr															
WDTRIS RO	0x010	WDTRIS															
WDTMIS RO	0x014	WDTMIS															
WDTTEST R/W	0x418	STALL															
WDTLOCK R/W	0xC00	WDTLock															
WDTRpid4 RO	0xFD0	PID4															
WDTRpid5 RO	0xFD4	PID5															
WDTRpid6 RO	0xFD8	PID6															

Name	Offset	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	
WDTPRPHD7 RO	0xFDC																PID7	
WDTPRPHD0 RO	0xFE0																PID0	
WDTPRPHD1 RO	0xFE4																PID1	
WDTPRPHD2 RO	0xFE8																PID2	
WDTPRPHD3 RO	0xFEC																PID3	
WDTPCID0 RO	0xFF0																CID0	
WDTPCID1 RO	0xFF4																CID1	
WDTPCID2 RO	0xFF8																CID2	
WDTPCID3 RO	0xFFC																CID3	
Analog-to-Digital Converter (ADC)																		
Base: 0x4003.8000																		
ADCACTSS R/W	0x000														ASEN3	ASEN2	ASEN1	ASEN0
ADCRIS RO	0x004														INR3	INR2	INR1	INR0
ADCIM R/W	0x008														MASK3	MASK2	MASK1	MASK0
ADCISC R/W1C	0x00C														IN3	IN2	IN1	IN0
ADCOSTAT R/W1C	0x010														OV3	OV2	OV1	OV0
ADCEMUX R/W	0x014																	
ADCUSTAT R/W1C	0x018														UV3	UV2	UV1	UV0
ADCSSPRI R/W	0x020																	
ADCPSSI	0x028																	

Name	Offset	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
WO														SS3	SS2	SS1	SS0
ADCSAC	0x030																AVG
ADCSMUX0	0x040			MUX7				MUX6				MUX5				MUX4	
				MUX3				MUX2				MUX1				MUX0	
ADCSCTL0	0x044	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
ADCSFF0	0x048																DATA
ADCSFF01	0x068																DATA
ADCSFF02	0x088																DATA
ADCSFSTAT0	0x04C				FULL				EMPTY					HPTR			TPTR
ADCSFSTAT1	0x06C				FULL				EMPTY					HPTR			TPTR
ADCSFSTAT2	0x08C				FULL				EMPTY					HPTR			TPTR
ADCSMUX1	0x060			MUX3				MUX2				MUX1				MUX0	
ADCSCTL1	0x064	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
ADCSMUX2	0x080			MUX3				MUX2				MUX1				MUX0	
ADCSCTL2	0x084	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
ADCSMUX3	0x0A0																MUX0
ADCSCTL3	0x0A4													TS0	IE0	END0	D0
ADCSFF03	0x0A8																
ADCSFSTAT3	0x0AC																
ADCTMLB	0x100																
										CNT		CONT	DIFF	TS			MUX

Name	Offset	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
ADCTMLB R/W	0x100																LB
Universal Asynchronous Receivers/Transmitters (UARTs) Base: 0x4000.C000 Base: 0x4000.D000 Base: 0x4000.E000																	
UARTDR RO	0x000					OE	BE	PE	FE								DATA
UARTRSR/ UARTECR R/W	0x004													OE	BE	PE	FE
UARTRSR/ UARTECR R/W	0x004																DATA
UARTFR RO	0x018									TXFE	RXFF	TXFF	RXFE	BUSY			
UARTILPR R/W	0x020																ILPDVSR
UARTIBRD R/W	0x024																DIVINT
UARTFBRD R/W	0x028																DIVFRAC
UARTLCRH R/W	0x02C									SPS	WLEN	FEN	STP2	EPS	PEN	BRK	
UARTCTL R/W	0x030							RXE	TXE	LBE					SIRLP	SIREN	UARTEN
UARTIFLS R/W	0x034																RXIFLSEL TXIFLSEL
UARTIM R/W	0x038						OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM				
UARTRIS RO	0x03C						OERIS	BERIS	PERIS	FERIS	RTRIS	TXRIS	RXRIS				
UARTMIS RO	0x040						OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS				
UARTICR W1C	0x044						OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC				
UARTPID4 RO	0xFD0																PID4
UARTPID5 RO	0xFD4																

Name	Offset	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				
RO																		PID5			
UARTPaID6 RO	0xFD8																	PID6			
UARTPaID7 RO	0xFDC																	PID7			
UARTPaID0 RO	0xFE0																	PID0			
UARTPaID1 RO	0xFE4																	PID1			
UARTPaID2 RO	0xFE8																	PID2			
UARTPaID3 RO	0xFEC																	PID3			
UARTPaID0 RO	0xFF0																	CID0			
UARTPaID1 RO	0xFF4																	CID1			
UARTPaID2 RO	0xFF8																	CID2			
UARTPaID3 RO	0xFFC																	CID3			
Synchronous Serial Interface (SSI)																					
Base: 0x4000.8000																					
SSICR0 R/W	0x000						SCR				SPH	SPO			FRF			DSS			
SSICR1 R/W	0x004														SOD	MS	SSE	LBM			
SSIDR R/W	0x008																				
SSISR RO	0x00C													BSY	RFF	RNE	TNF	TFE			
SSICPSR R/W	0x010																	CPSDVSR			
SSIIM R/W	0x014																	TXIM	RXIM	RTIM	RORIM
SSIRIS RO	0x018																	TXRIS	RXRIS	RTRIS	RORRIS

Name	Offset	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		
SSIMIS RO	0x01C														TXMIS	RXMIS	RTMIS	RORMIS	
SSIICR W1C	0x020																RTIC	RORIC	
SSIPerphID4 RO	0xFD0																	PID4	
SSIPerphID5 RO	0xFD4																	PID5	
SSIPerphID6 RO	0xFD8																	PID6	
SSIPerphID7 RO	0xFDC																	PID7	
SSIPerphID0 RO	0xFE0																	PID0	
SSIPerphID1 RO	0xFE4																	PID1	
SSIPerphID2 RO	0xFE8																	PID2	
SSIPerphID3 RO	0xFEC																	PID3	
SSIPCellID0 RO	0xFF0																	CID0	
SSIPCellID1 RO	0xFF4																	CID1	
SSIPCellID2 RO	0xFF8																	CID2	
SSIPCellID3 RO	0xFFC																	CID3	
Inter-Integrated Circuit (I ² C) Interface																			
Base: 0x4002.0000																			
Base: 0x4002.0800																			
Base: 0x4002.1000																			
Base: 0x4001.1800																			
I2CMSA R/W	0x000																	SA	R/S
I2CMCS R/W	0x004										BUSBSY	IDLE	ARBLST	DATAACK	ADRACK	ERROR		BUSY	
I2CMCS	0x004																		

Name	Offset	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
R/W														ACK	STOP	START	RUN
I2CMDR R/W	0x008													DATA			
I2CMTPR R/W	0x00C																
I2CMIMR R/W	0x010																IM
I2CMRIS RO	0x014																RIS
I2CMMIS RO	0x018																MIS
I2CMICR WO	0x01C																IC
I2CMCR R/W	0x020											SFE	MFE				LPBK
I2CSOAR R/W	0x000													OAR			
I2CSCSR RO	0x004														FBR	TREQ	RREQ
I2CSCSR RO	0x004																DA
I2CSDR R/W	0x008													DATA			
I2CSIMR R/W	0x00C																IM
I2CSRIS RO	0x010																RIS
I2CSMIS RO	0x014																MIS
I2CSICR WO	0x018																IC
Controller Area Network (CAN) Module																	
Base: 0x4004.0000																	
CANCTL R/W	0x000									Test	CCE	DAR		EIE	SIE	IE	INIT
CANSTS R/W	0x004									BOff	EWarn	EPass	RxOK	TxOK			LEC

Name	Offset	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
CANERRRO	0x008																
		RP	REC						TEC								
CANBITR/W	0x00C																
			TSeg2			TSeg1			SJW			BRP					
CANINTRO	0x010																
			IntId														
CANTSTR/W	0x014																
									Rx	Tx	LBack	Silent	Basic				
CANBRPER/W	0x018																
			BRPE														
CANF1CRQR/W	0x020	Busy															
			MNUM														
CANF2CRQR/W	0x080	Busy															
			MNUM														
CANF1MSKR/W	0x024																
									WRNRD	Mask	Arb	Control	ClIntPnd	TRnDt	DataA	DataB	
CANF2MSKR/W	0x084																
									WRNRD	Mask	Arb	Control	ClIntPnd	TRnDt	DataA	DataB	
CANF1MSK1R/W	0x028																
			Msk														
CANF2MSK1R/W	0x088																
			Msk														
CANF1MSK2R/W	0x02C	MXtd	MDir														
			Msk														
CANF2MSK2R/W	0x08C	MXtd	MDir														
			Msk														
CANF1ARB1R/W	0x030																
			ID														
CANF2ARB1R/W	0x090																
			ID														
CANF1ARB2R/W	0x034	MsgVal	Xtd	Dir													
			ID														
CANF2ARB2R/W	0x094	MsgVal	Xtd	Dir													
			ID														
CANF1MCTLR/W	0x038	NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst	EoB							
			DLC														
CANF2MCTL	0x098																

Name	Offset	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	
R/W		NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst	EoB					DLC			
CANIF1DA1 R/W	0x03C									Data								
CANIF2DA1 R/W	0x09C									Data								
CANIF1DA2 R/W	0x040									Data								
CANIF2DA2 R/W	0x0A0									Data								
CANIF1DB1 R/W	0x044									Data								
CANIF2DB1 R/W	0x0A4									Data								
CANIF1DB2 R/W	0x048									Data								
CANIF2DB2 R/W	0x0A8									Data								
CANTXRQ1 RO	0x100									TxRqst								
CANTXRQ2 RO	0x104									TxRqst								
CANNMDA1 RO	0x120									NewDat								
CANNMDA2 RO	0x124									NewDat								
CANMSGINT RO	0x140									IntPnd								
CANMSGINT RO	0x144									IntPnd								
CANMSGVAL RO	0x160									MsgVal								
CANMSGVAL RO	0x164									MsgVal								
Ethernet Controller Base: 0x4004.8000																		
MACRIS RO	0x000											PHYINT	MDINT	RXER	FOV	TXEMP	TXER	RXINT

Name	Offset	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
MACIACK W1C	0x000											MDINT	RXER	FOV	TXEMP	TXER	RXINT
MACIM R/W	0x004									PHYNTM	MDINTM	RXERM	FOVM	TXEMPM	TXERM	RXINTM	
MACRCTL R/W	0x008												RSTFIFO	BADCRC	PRMS	AMUL	RXEN
MACTCTL R/W	0x00C												DUPLEX		CRC	PADEN	TXEN
MACDATA R/W	0x010																
MACDATA R/W	0x010																
MACIA0 R/W	0x014																
MACIA1 R/W	0x018																
MACTHR R/W	0x01C																
MACMCTL R/W	0x020																
MACMDV R/W	0x024																
MACMADD RO	0x028																
MACMTXD R/W	0x02C																
MACMRXD R/W	0x030																
MACNP RO	0x034																
MACTR R/W	0x038																
MR0 R/W	0x00	RESET	LOOPBK	SPEEDSL	ANEGEN	PWRDN	ISO	RANEG	DUPLEX	COLT							
MR1 RO	0x01		100X_F	100X_H	10T_F	10T_H					MFPS	ANEGC	RFAULT	ANEGA	LINK	JAB	EXTD
MR2	0x02																

Name	Offset	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	
RO		OUI[21:6]																
MR3 RO	0x03	OUI[5:0]					MN					RN						
MR4 R/W	0x04	NP		RF					A3	A2	A1	A0	S[4:0]					
MR5 RO	0x05	NP	ACK	RF	A[7:0]							S[4:0]						
MR6 RO	0x06												PDF	LPNPA		PRX	LPANEGA	
MR16 R/W	0x10	RPTR	INPOL		TXHIM	SQEI	NL10				APOL	RVSPOL				PCSBP	RXCC	
MR17 R/W	0x11	JABBER_E	RXER_IE	PRX_IE	PDF_IE	LPACK_IE	LSCHG_IE	RFALTE	ANEGOME	JABBER_INT	RXER_INT	PRX_INT	PDF_INT	LPACK_INT	LSCHG_INT	RFALTE_INT	ANEGOMINT	
MR18 RO	0x12				ANEGF	DPLX	RATE	RXSD	RX_LOCK									
MR19 R/W	0x13	TXO[1:0]																
MR23 R/W	0x17										LED1[3:0]			LED0[3:0]				
MR24 R/W	0x18										FD_MODE	AUTO_SW	MDIX	MDIX_CM	MDIX_SD			
Analog Comparators																		
Base: 0x4003.C000																		
ACMIS R/W1C	0x00														IN2	IN1	IN0	
ACRIS RO	0x04														IN2	IN1	IN0	
ACINTEN R/W	0x08														IN2	IN1	IN0	
ACREFCTL R/W	0x10							EN	RNG						VREF			
ACSTAT0 RO	0x20																OVAL	
ACSTAT1 RO	0x40																OVAL	
ACSTAT2 RO	0x60																OVAL	

Name	Offset	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
ACCTL0 R/W	0x24																
						TOEN	ASRCP			TSLVAL	TSEN	ISLVAL	ISEN	CINV			
ACCTL1 R/W	0x44																
						TOEN	ASRCP			TSLVAL	TSEN	ISLVAL	ISEN	CINV			
ACCTL2 R/W	0x64																
						TOEN	ASRCP			TSLVAL	TSEN	ISLVAL	ISEN	CINV			

DRAFT