TEXAS INSTRUMENTS-ADVANCE INFORMATION

Stellaris® LM3S1W16 Microcontroller

DATA SHEET

DS-LM3S1W16-6419

Copyright

Texas Instruments Incorporated
108 Wild Basin, Suite 350
Austin, TX 78746
Main: +1-512-279-8800
Fax: +1-512-279-8879
http://www.luminarymicro.com

Table of Contents

List of Figures

List of Tables

List of Registers

Revision History

The revision history table notes changes made between the indicated revisions of the LM3S1W16 data sheet.

Table 1. Revision History

Date	Revision	Description
May 2009	5285	Started tracking revision history.
June 2009	5779	■ In System Control chapter, clarified power-on reset and external reset pin descriptions in "Reset Sources" section. ■ Added missing comparator output pin bits to **DC3** register; reset value changed as well. ■ Clarified explanation of nonvolatile register programming in Internal Memory chapter. ■ Added explanation of reset value to **FMPRE0/1/2/3**, **FMPPE0/1/2/3**, **USER_DBG**, and **USER_REG0** registers. ■ In Request Type Support table in DMA chapter, corrected general-purpose timer row. ■ In General-Purpose Timers chapter, clarified DMA operation. ■ Added table "Preliminary Current Consumption" to Characteristics chapter. ■ Corrected Nom and Max values in "Hibernation Detailed Current Specifications" table. ■ Corrected Nom and Max values in EPI Characteristics table. ■ Added "CSn to output invalid" parameter to EPI table "EPI Host-Bus 8 and Host-Bus 16 Interface Characteristics" and figure "Host-Bus 8/16 Mode Read Timing". ■ Corrected INL, DNL, OFF and GAIN values in ADC Characteristics table. ■ Updated ROM DriverLib appendix with RevC0 functions. ■ Updated part ordering numbers. ■ Additional minor data sheet clarifications and corrections.

Table 1. Revision History *(continued)*

Date	Revision	Description
July 2009	5930	■ Corrected values for `MAXADC0SPD` and `MAXADC1SPD` bits in **DC1**, **RCGC0**, **SCGC0**, and **DCGC0** registers. ■ Corrected figure "TI Synchronous Serial Frame Format (Single Transfer)". ■ Changed `HIB` pin from type TTL to type OD. ■ Made a number of corrections to the Electrical Characteristics chapter: – Deleted V_{BAT} and V_{REFA} parameters from and added footnotes to Recommended DC Operating Conditions table. – Modified Hibernation Module DC Characteristics table. – Deleted Nominal and Maximum Current Specifications section. – Deleted SDRAM Read Command Timing, SDRAM Write Command Timing, SDRAM Write Burst Timing, SDRAM Precharge Command Timing and SDRAM CAS Latency Timing figures and replaced with SDRAM Read Timing and SDRAM Write Timing figures. – Modified Host-Bus 8/16 Mode Write Timing figure. – Modified General-Purpose Mode Read and Write Timing figure. – Major changes to ADC Characteristics tables, including adding additonal tables and diagram. ■ Corrected ordering part numbers. ■ Additional minor data sheet clarifications and corrections.

Table 1. Revision History *(continued)*

Date	Revision	Description
October 2009	6419	■ Released new 1000, 3000, 5000 and 9000 series Stellaris® devices. ■ The IDCODE value was corrected to be 0x4BA0.0477. ■ Clarified that the `NMISET` bit in the **ICSR** register in the NVIC is also a source for NMI. ■ Clarified the use of the LDO. ■ To clarify clock operation, reorganized clocking section, changed the `USEFRACT` bit to the `DIV400` bit and the `FRACT` bit to the `SYSDIV2LSB` bit in the **RCC2** register, added tables, and rewrote descriptions. ■ Corrected bit description of the `DSDIVORIDE` field in the **DSLPCLKCFG** register. ■ Removed the **DSFLASHCFG** register at System Control offset 0x14C as it does not function correctly. ■ Removed the `MAXADC1SPD` and `MAXADC0SPD` fields from the **DCGC0** as they have no function in deep-sleep mode. ■ Corrected address offsets for the **Flash Write Buffer (FWBn)** registers. ■ Added **Flash Control (FCTL)** register at Internal memory offset 0x0F8 to help control frequent power cycling when hibernation is not used. ■ Changed the name of the EPI channels for clarification: EPI0_TX became EPI0_WFIFO and EPI0_RX became EPI0_NBRFIFO. This change was also made in the DC7 bit descriptions. ■ Removed the **DMACHIS** register at DMA module offset 0x504 as it does not function correctly. ■ Corrected alternate channel assignments for the µDMA controller. ■ Major improvements to the EPI chapter. ■ **EPISDRAMCFG2** register was deleted as its function is not needed. ■ Clarified PWM source for ADC triggering ■ Changed SSI set up and hold times to be expressed in system clocks, not ns. ■ Updated Electrical Characteristics chapter with latest data. Changes were made to Hibernation, ADC and EPI content. ■ Additional minor data sheet clarifications and corrections.

About This Document

This data sheet provides reference information for the LM3S1W16 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 Stellaris® 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*
- *Stellaris® Peripheral Driver Library User's Guide*
- *Stellaris® ROM User’s Guide*

The following related documents are also referenced:

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

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

Documentation Conventions

This document uses the conventions shown in Table 2 on page 25.

Table 2. Documentation Conventions

Notation	Meaning
General Register Notation	
REGISTER	APB registers are indicated in uppercase bold. For example, **PBORCTL** is the Power-On and Brown-Out Reset Control register. If a register name contains a lowercase n, it represents more than one register. For example, **SRCRn** represents any (or all) of the three Software Reset Control registers: **SRCR0, SRCR1** , and **SRCR2**.
bit	A single bit in a register.
bit field	Two or more consecutive and related bits.
offset 0x*nnn*	A hexadecimal increment to a register's address, relative to that module's base address as specified in “Memory Map” on page 60.

Table 2. Documentation Conventions *(continued)*

Notation	Meaning
Register *N*	Registers are numbered consecutively throughout the document to aid in referencing them. The register number has no meaning to software.
reserved	Register bits marked *reserved* 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.
R/W1S	Software can read or write a 1 to this field. A write of a 0 to a R/W1S bit does not affect the bit value in the register.
W1C	Software can write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged. A read of the register returns no meaningful data. This register is typically used to clear the corresponding bit in an interrupt register.
WO	Only a write by software is valid; a read of the register returns no meaningful data.
Register Bit/Field Reset Value	This value in the register bit diagram shows the bit/field value after any reset, unless noted.
0	Bit cleared to 0 on chip reset.
1	Bit set to 1 on chip reset.
-	Nondeterministic.
Pin/Signal Notation	
[ ]	Pin alternate function; a pin defaults to the signal without the brackets.
pin	Refers to the physical connection on the package.
signal	Refers to the electrical signal encoding of a pin.
assert a signal	Change the value of the signal from the logically False state to the logically True state. For active High signals, the asserted signal value is 1 (High); for active Low signals, the asserted signal value is 0 (Low). The active polarity (High or Low) is defined by the signal name (see `SIGNAL` and $\overline{\texttt{SIGNAL}}$ below).
deassert a signal	Change the value of the signal from the logically True state to the logically False state.
$\overline{\texttt{SIGNAL}}$	Signal names are in uppercase and in the Courier font. An overbar on a signal name indicates that it is active Low. To assert $\overline{\texttt{SIGNAL}}$ is to drive it Low; to deassert $\overline{\texttt{SIGNAL}}$ is to drive it High.
`SIGNAL`	Signal names are in uppercase and in the Courier font. An active High signal has no overbar. To assert `SIGNAL` is to drive it High; to deassert `SIGNAL` is to drive it Low.
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.

Table 2. Documentation Conventions *(continued)*

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

1 Architectural Overview

Texas Instruments is the industry leader in bringing 32-bit capabilities and the full benefits of ARM® Cortex-M3™-based microcontrollers to the broadest reach of the microcontroller market. For current users of 8- and 16-bit MCUs, Stellaris® with Cortex-M3 offers a direct path to the strongest ecosystem of development tools, software and knowledge in the industry. Designers who migrate to Stellaris® benefit from great tools, small code footprint and outstanding performance. Even more important, designers can enter the ARM ecosystem with full confidence in a compatible roadmap from $1 to 1 GHz. For users of current 32-bit MCUs, the Stellaris® family offers the industry's first implementation of Cortex-M3 and the Thumb-2 instruction set. With blazingly-fast responsiveness, Thumb-2 technology combines both 16-bit and 32-bit instructions to deliver the best balance of code density and performance. Thumb-2 uses 26 percent less memory than pure 32-bit code to reduce system cost while delivering 25 percent better performance. The Texas Instruments Stellaris® family of microcontrollers—the first ARM® Cortex™-M3 based controllers—brings high-performance 32-bit computing to cost-sensitive embedded microcontroller applications. These pioneering parts deliver customers 32-bit performance at a cost equivalent to legacy 8- and 16-bit devices, all in a package with a small footprint.

The LM3S1W16 microcontroller has the following features:

- ARM® Cortex™-M3 Processor Core
  - 50-MHz operation; 60 DMIPS performance
  - ARM Cortex SysTick Timer
  - Nested Vectored Interrupt Controller (NVIC)
- On-Chip Memory
  - 32 KB single-cycle Flash memory
  - 8 KB single-cycle SRAM
  - Internal ROM loaded with StellarisWare® software:
    - Stellaris® Peripheral Driver Library
    - Stellaris® Boot Loader
- Advanced Serial Integration
  - Three UARTs with IrDA and ISO 7816 support
  - Two I^{2}C modules
  - Two Synchronous Serial Interface modules (SSI)
- System Integration
  - Direct Memory Access Controller (DMA)
  - System control and clocks including on-chip precision 16-MHz oscillator

  - Three 32-bit timers (up to six 16-bit)
  - Six Capture Compare PWM pins (CCP)
  - Lower-power battery-backed hibernation module
  - Real-Time Clock
  - Two Watchdog Timers
    - One timer runs off the main oscillator
    - One timer runs off the precision internal oscillator
  - Up to 33 GPIOs, depending on configuration
    - Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
    - Independently configurable to 2, 4 or 8 mA drive capability
    - Up to 4 GPIOs can have 18 mA drive capability
- Analog
  - 10-bit Analog-to-Digital Converter (ADC) with eight analog input channels and sample rate of one million samples/second
  - Two analog comparators
  - Eight digital comparators
  - On-chip voltage regulator
- JTAG and ARM Serial Wire Debug (SWD)
- 64-pin LQFP package
- Industrial (-40°C to 85°C) Temperature Range

The Stellaris® LM3S1000 microcontrollers are perfect for cost-effective embedded control applications.

The LM3S1W16 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 LM3S1W16 microcontroller features a battery-backed Hibernation module to efficiently power down the LM3S1W16 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 LM3S1W16 microcontroller perfectly for battery applications.

In addition, the LM3S1W16 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 LM3S1W16 microcontroller is code-compatible to all members of the extensive Stellaris® family; providing flexibility to fit our customers' precise needs.

Texas Instruments offers a complete solution to get to market quickly, with evaluation and development boards, white papers and application notes, an easy-to-use peripheral driver library, and a strong support, sales, and distributor network. See “Ordering and Contact Information” on page 697 for ordering information for Stellaris® family devices.

1.1 Functional Overview

The following sections provide an overview of the features of the LM3S1W16 microcontroller. The page number in parentheses indicates where that feature is discussed in detail. Ordering and support information can be found in “Ordering and Contact Information” on page 697.

1.1.1 ARM Cortex™-M3

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

1.1.1.1 Processor Core (see page 47)

All members of the Stellaris® product family, including the LM3S1W16 microcontroller, are designed around an ARM Cortex™-M3 processor core. The ARM Cortex-M3 processor provides the core for a high-performance, low-cost platform that meets the needs of minimal memory implementation, reduced pin count, and low power consumption, while delivering outstanding computational performance and exceptional system response to interrupts.

- 32-bit ARM® Cortex™-M3 v7M architecture optimized for small-footprint embedded applications
- Outstanding processing performance combined with fast interrupt handling
- Thumb-2 mixed 16-/32-bit instruction set, delivers the high performance expected of a 32-bit ARM core in a compact memory size usually associated with 8- and 16-bit devices; typically in the range of a few kilobytes of memory for microcontroller-class applications
  - Single-cycle multiply instruction and hardware divide
  - Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
  - Unaligned data access, enabling data to be efficiently packed into memory
- Fast code execution permits slower processor clock or increases sleep mode time
- Harvard architecture characterized by separate buses for instruction and data
- Efficient processor core, system and memories
- Hardware division and fast multiplier
- Deterministic, high-performance interrupt handling for time-critical applications
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionality
- Enhanced system debug with extensive breakpoint and trace capabilities

- Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging and tracing
- Migration from the ARM7™ processor family for better performance and power efficiency
- Optimized for single-cycle Flash memory usage
- Ultra-low power consumption with integrated sleep mode
- 50-MHz operation
- 1.25 DMIPS/MHz

"ARM Cortex-M3 Processor Core" on page 47 provides an overview of the ARM core; the core is detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

1.1.1.2 System Timer (SysTick) (see page 57)

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

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

1.1.1.3 Nested Vectored Interrupt Controller (NVIC) (see page 63)

The LM3S1W16 controller includes the ARM Nested Vectored Interrupt Controller (NVIC). The NVIC and Cortex-M3 prioritize and handle all exceptions in Handler Mode. The processor state is automatically stored to the stack on an exception and automatically restored from the stack at the end of the Interrupt Service Routine (ISR). The interrupt vector is fetched in parallel to the state saving, enabling efficient interrupt entry. The processor supports tail-chaining, meaning that back-to-back interrupts can be performed without the overhead of state saving and restoration. Software can set eight priority levels on 7 exceptions (system handlers) and 30 interrupts.

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

"Interrupts" on page 63 provides an overview of the NVIC controller and the interrupt map. Exceptions and interrupts are detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

1.1.2 On-Chip Memory

The following sections describe the on-chip memory modules.

1.1.2.1 SRAM (see page 195)

The LM3S1W16 microcontroller provides 8 KB of single-cycle on-chip SRAM. The internal SRAM of the Stellaris® devices is located at offset 0x2000.0000 of the device memory map.

Because read-modify-write (RMW) operations are very time consuming, ARM has introduced *bit-banding* technology in the new Cortex-M3 processor. With a bit-band-enabled processor, certain regions in the memory map (SRAM and peripheral space) can use address aliases to access individual bits in a single, atomic operation.

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

1.1.2.2 Flash Memory (see page 195)

The LM3S1W16 microcontroller provides 32 KB of single-cycle on-chip Flash memory. The Flash memory is organized as a set of 2-KB blocks that can be individually erased. Erasing a block causes the entire contents of the block to be reset to all 1s. These blocks are paired into a set of 2-KB blocks that can be individually protected. The blocks can be marked as read-only or execute-only, providing different levels of code protection. Read-only blocks cannot be erased or programmed, protecting the contents of those blocks from being modified. Execute-only blocks cannot be erased or programmed, and can only be read by the controller instruction fetch mechanism, protecting the contents of those blocks from being read by either the controller or by a debugger.

1.1.2.3 ROM (see page 663)

The LM3S1W16 ROM is preprogrammed with the following software and programs:

- Stellaris® Peripheral Driver Library
- Stellaris® Boot Loader

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

1.1.3 Serial Communications Peripherals

The LM3S1W16 controller supports both asynchronous and synchronous serial communications with:

- Three UARTs with IrDA and ISO 7816 support
- Two I^{2}C modules
- Two Synchronous Serial Interface Modules (SSI)

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

1.1.3.1 UART (see page 478)

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 LM3S1W16 controller includes three fully programmable 16C550-type UARTs. Although the functionality is similar to a 16C550 UART, this UART design is not register compatible. The UART can generate individually masked interrupts from the Rx, Tx, modem status, and error conditions. The module generates a single combined interrupt when any of the interrupts are asserted and are unmasked.

The three UARTs have the following features:

- Programmable baud-rate generator allowing speeds up to 3.125 Mbps for regular speed (divide by 16) and 6.25 Mbps for high speed (divide by 8)
- Separate 16x8 transmit (TX) and receive (RX) FIFOs to reduce CPU interrupt service loading
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
- Standard asynchronous communication bits for start, stop, and parity
- 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 μs) bit durations
  - Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration
- Support for communication with ISO 7816 smart cards
- LIN protocol support
- Standard FIFO-level and End-of-Transmission interrupts

- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Separate channels for transmit and receive
  - Receive single request asserted when data is in the FIFO; burst request asserted at programmed FIFO level
  - Transmit single request asserted when there is space in the FIFO; burst request asserted at programmed FIFO level

1.1.3.2 I^{2}C (see page 577)

The Inter-Integrated Circuit (I^{2}C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL). The I^{2}C bus interfaces to external I^{2}C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I^{2}C bus may also be used for system testing and diagnostic purposes in product development and manufacture.

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

The LM3S1W16 controller includes two I^{2}C modules with the following features:

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

1.1.3.3 SSI (see page 535)

Synchronous Serial Interface (SSI) is a four-wire bi-directional communications interface that converts data between parallel and serial. The SSI module performs serial-to-parallel conversion on data received from a peripheral device, and parallel-to-serial conversion on data transmitted to a peripheral device. The SSI module can be configured as either a master or slave device. As a slave device, the SSI module can also be configured to disable its output, which allows a master device to be coupled with multiple slave devices. The TX and RX paths are buffered with separate internal FIFOs.

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

The LM3S1W16 controller includes two SSI modules with the following features:

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

1.1.4 System Integration

The LM3S1W16 controller provides a variety of standard system functions integrated into the device, including:

- Micro Direct Memory Access Controller (μDMA)
- System control and clocks including on-chip precision 16-MHz oscillator
- ARM Cortex SysTick Timer
- Three 32-bit timers (up to six 16-bit)
- Six Capture Compare PWM pins (CCP)
- Lower-power battery-backed hibernation module
- Real-Time Clock
- Two Watchdog Timers

- Up to 33 GPIOs, depending on configuration
  - Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
  - Independently configurable to 2, 4 or 8 mA drive capability
  - Up to 4 GPIOs can have 18 mA drive capability

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

1.1.4.1 Direct Memory Access (see page 227)

The LM3S1W16 microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (μDMA). The μDMA controller provides a way to offload data transfer tasks from the Cortex-M3 processor, allowing for more efficient use of the processor and the available bus bandwidth. The μDMA controller can perform transfers between memory and peripherals. It has dedicated channels for each supported on-chip module and can be programmed to automatically perform transfers between peripherals and memory as the peripheral is ready to transfer more data. The μDMA controller provides the following features:

- ARM PrimeCell® 32-channel configurable μDMA controller
- Support for memory-to-memory, memory-to-peripheral, and peripheral-to-memory in multiple transfer modes
  - Basic for simple transfer scenarios
  - Ping-pong for continuous data flow
  - Scatter-gather for a programmable list of arbitrary transfers initiated from a single request
- Highly flexible and configurable channel operation
  - Independently configured and operated channels
  - Dedicated channels for supported on-chip modules: GP Timer, UART, ADC, SSI
  - Alternate channel assignments
  - One channel each for receive and transmit path for bidirectional modules
  - Dedicated channel for software-initiated transfers
  - Per-channel configurable bus arbitration scheme
  - Optional software-initiated requests for any channel
- Two levels of priority
- Design optimizations for improved bus access performance between μDMA controller and the processor core
  - μDMA controller access is subordinate to core access
  - RAM striping
  - Peripheral bus segmentation

- Data sizes of 8, 16, and 32 bits
- Transfer size is programmable in binary steps from 1 to 1024
- Source and destination address increment size of byte, half-word, word, or no increment
- Maskable peripheral requests
- Interrupt on transfer completion, with a separate interrupt per channel

1.1.4.2 System Control and Clocks (see page 78)

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

- Device identification information: version, part number, SRAM size, Flash memory size, and so on
- Power control
  - On-chip fixed Low Drop-Out (LDO) voltage regulator
  - Hibernation module handles the power-up/down 3.3 V sequencing and control for the core digital logic and analog circuits
  - Low-power options for microcontroller: Sleep and Deep-sleep modes with clock gating
  - Low-power options for on-chip modules: software controls shutdown of individual peripherals and memory
  - 3.3-V supply brown-out detection and reporting via interrupt or reset
- Multiple clock sources for microcontroller system clock
  - Precision Oscillator (PIOSC): on-chip resource providing a 16 MHz ±1% frequency at room temperature
    - 16 MHz ±3% across temperature
    - Can be recalibrated with 7-bit trim resolution
    - Software power down control for low power modes
  - Main Oscillator (MOSC): a frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the `OSC0` input pin, or an external crystal is connected across the `OSC0` input and `OSC1` output pins.
    - External oscillator used with or without on-chip PLL: select supported frequencies from 1 MHz to 16.384 MHz.
    - External crystal: from DC to maximum device speed
  - Internal 30-kHz Oscillator: on chip resource providing a 30 kHz ± 50% frequency, used during power-saving modes

  - Hibernation Module clock source: eliminates need for additional crystal for main clock source
    - 32.768-kHz external oscillator
    - 4.194304-MHz external crystal
- Flexible reset sources
  - Power-on reset (POR)
  - Reset pin assertion
  - Brown-out reset (BOR) detector alerts to system power drops
  - Software reset
  - Watchdog timer reset
  - MOSC failure

1.1.4.3 Three Programmable Timers (see page 338)

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

The General-Purpose Timer Module (GPTM) contains three GPTM blocks with the following functional options:

- Count up or down
- 16- or 32-bit programmable one-shot timer
- 16- or 32-bit programmable periodic timer
- 16-bit general-purpose timer with an 8-bit prescaler
- 32-bit Real-Time Clock (RTC) when using an external 32.768-KHz clock as the input
- Six Capture Compare PWM pins (CCP)
- Daisy chaining of timer modules to allow a single timer to initiate multiple timing events
- ADC event trigger
- User-enabled stalling when the controller asserts CPU Halt flag during debug (excluding RTC mode)
- 16-bit input-edge count- or time-capture modes
- 16-bit PWM mode with software-programmable output inversion of the PWM signal
- Ability to determine the elapsed time between the assertion of the timer interrupt and entry into the interrupt service routine.
- Efficient transfers using Micro Direct Memory Access Controller (μDMA)

  - Dedicated channel for each timer
  - Burst request generated on timer interrupt

1.1.4.4 CCP Pins (see page 343)

Capture Compare PWM pins (CCP) can be used by the General-Purpose Timer Module to time/count external events using the CCP pin as an input. Alternatively, the GPTM can generate a simple PWM output on the CCP pin.

The LM3S1W16 microcontroller includes six Capture Compare PWM pins (CCP) that can be programmed to operate in the following modes:

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

1.1.4.5 Hibernation Module (see page 168)

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

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

1.1.4.6 Watchdog Timers (see page 381)

A watchdog timer is used to regain control when a system has failed due to a software error or to the failure of an external device to respond in the expected way. The Stellaris® Watchdog Timer can generate a nonmaskable interrupt (NMI) or a reset when a time-out value is reached. In addition, the Watchdog Timer is ARM FiRM-compliant and can be configured to generate an interrupt to the controller on its first time-out, and to generate a reset signal on its second time-out. Once the Watchdog Timer has been configured, the lock register can be written to prevent the timer configuration from being inadvertently altered.

The LM3S1W16 microcontroller has two Watchdog Timer modules: Watchdog Timer 0 uses the system clock for its timer clock; Watchdog Timer 1 uses the PIOSC as its timer clock. The Stellaris® Watchdog Timer module has the following features:

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

1.1.4.7 Programmable GPIOs (see page 285)

General-purpose input/output (GPIO) pins offer flexibility for a variety of connections. The Stellaris® GPIO module is comprised of five physical GPIO blocks, each corresponding to an individual GPIO port. The GPIO module is FiRM-compliant (compliant to the ARM Foundation IP for Real-Time Microcontrollers specification) and supports 0-33 programmable input/output pins. The number of GPIOs available depends on the peripherals being used (see “Signal Tables” on page 627 for the signals available to each GPIO pin).

- Up to 33 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 5-V-tolerant input/outputs
- Fast toggle capable of a change every two clock cycles
- Two means of port access: either Advanced High-Performance Bus (AHB) with better back-to-back access performance, or the legacy Advanced Peripheral Bus (APB) for backwards-compatibility with existing code
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
- Bit masking in both read and write operations through address lines
- Can be used to initiate an ADC sample sequence
- Pins configured as digital inputs are Schmitt-triggered

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

1.1.5 Analog

The LM3S1W16 controller provides analog functions integrated into the device, including:

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

The following provides more detail on these analog functions.

1.1.5.1 ADC (see page 406)

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. Four buffered sample sequencers allow rapid sampling of up to eight analog input sources without controller intervention. Each sample sequencer provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequencer priority. A digital comparator function is included that allows the conversion value to be diverted to a comparison unit that provides eight digital comparators.

with the following features:

- Eight analog input channels
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Sample rate of one million samples/second
- Flexible, configurable analog-to-digital conversion
- Four programmable sample conversion sequencers from one to eight entries long, with corresponding conversion result FIFOs
- Flexible trigger control
  - Controller (software)

  - Timers
  - Analog Comparators
  - GPIO
- Hardware averaging of up to 64 samples for improved accuracy
- Digital comparison unit providing eight digital comparators
- Converter uses an internal 3-V reference or an external reference
- Power and ground for the analog circuitry is separate from the digital power and ground
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Dedicated channel for each sample sequencer
  - Burst request asserted when interrupt is triggered

1.1.5.2 Analog Comparators (see page 614)

An analog comparator is a peripheral that compares two analog voltages and provides a logical output that signals the comparison result. The LM3S1W16 microcontroller provides two independent integrated analog comparators that can be configured to drive an output or generate an interrupt or ADC event.

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

The LM3S1W16 microcontroller provides two independent integrated analog comparators with the following functions:

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

1.1.6 JTAG and ARM Serial Wire Debug (see page 66)

The Joint Test Action Group (JTAG) port is an IEEE standard that defines a Test Access Port and Boundary Scan Architecture for digital integrated circuits and provides a standardized serial interface for controlling the associated test logic. The TAP, Instruction Register (IR), and Data Registers (DR) can be used to test the interconnections of assembled printed circuit boards and obtain manufacturing information on the components. The JTAG Port also provides a means of accessing and controlling design-for-test features such as I/O pin observation and control, scan testing, and debugging. Texas Instruments replaces the ARM SW-DP and JTAG-DP with the ARM CoreSight™-compliant Serial Wire JTAG Debug Port (SWJ-DP) interface. The SWJ-DP interface combines the SWD and JTAG debug ports into one module providing all the normal JTAG debug and test functionality plus real-time access to system memory without halting the core or requiring any target resident code. See the *CoreSight™ Design Kit Technical Reference Manual* for details on SWJ-DP. The SWJ-DP interface has the following features:

- IEEE 1149.1-1990 compatible Test Access Port (TAP) controller
- Four-bit Instruction Register (IR) chain for storing JTAG instructions
- IEEE standard instructions: BYPASS, IDCODE, SAMPLE/PRELOAD, EXTEST and INTEST
- ARM additional instructions: APACC, DPACC and ABORT
- Integrated ARM Serial Wire Debug (SWD)
  - Serial Wire JTAG Debug Port (SWJ-DP)
  - Flash Patch and Breakpoint (FPB) unit for implementing breakpoints
  - Data Watchpoint and Trigger (DWT) unit for implementing watchpoints, trigger resources, and system profiling
  - Instrumentation Trace Macrocell (ITM) for support of printf style debugging
  - Trace Port Interface Unit (TPIU) for bridging to a Trace Port Analyzer

1.1.7 Packaging and Temperature

- Industrial-range 64-pin RoHS-compliant LQFP package

1.2 Target Applications

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

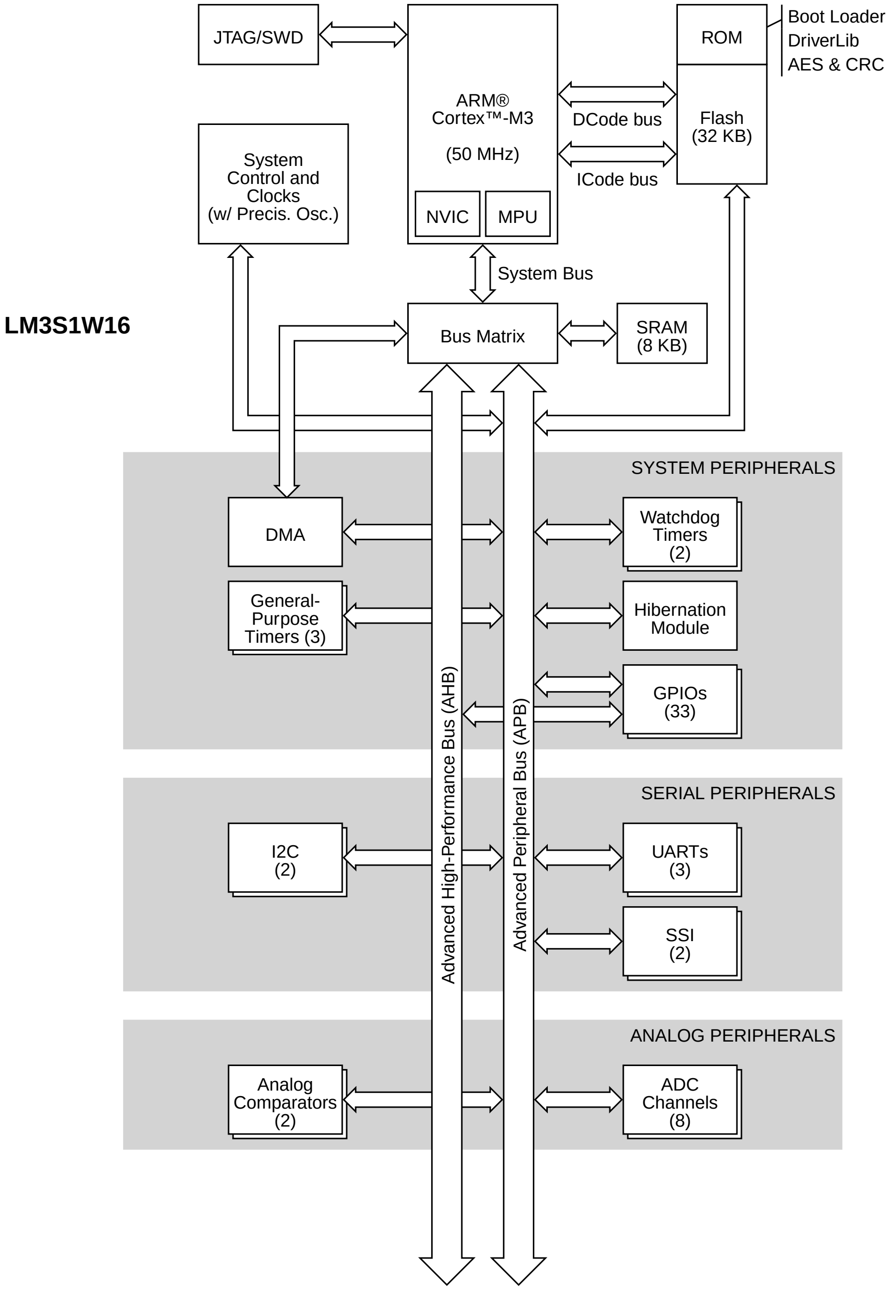
- Remote monitoring
- Electronic point-of-sale (POS) machines
- Test and measurement equipment
- Network appliances and switches
- Factory automation
- HVAC and building control
- Gaming equipment
- Motion control
- Medical instrumentation
- Fire and security
- Power and energy

- Transportation

1.3 High-Level Block Diagram

Figure 1-1 depicts the features on the Stellaris® LM3S1W16 microcontroller. Note that there are two on-chip buses that connect the core to the peripherals. The Advanced Peripheral Bus (APB) bus is the legacy bus. The Advanced High-Performance Bus (AHB) bus provides better back-to-back access performance than the APB bus.

Figure 1-1. Stellaris® LM3S1W16 Microcontroller High-Level Block Diagram



1.4 Additional Features

1.4.1 Memory Map (see page 60)

A memory map lists the location of instructions and data in memory. The memory map for the LM3S1W16 controller can be found in "Memory Map" on page 60. Register addresses are given as a hexadecimal increment, relative to the module's base address as shown in the memory map. The *ARM® Cortex™-M3 Technical Reference Manual* provides further information on the memory map.

1.4.2 Hardware Details

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

- "Pin Diagram" on page 626
- "Signal Tables" on page 627
- "Operating Characteristics" on page 642
- "Electrical Characteristics" on page 643
- "Package Information" on page 699

2 ARM Cortex-M3 Processor Core

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

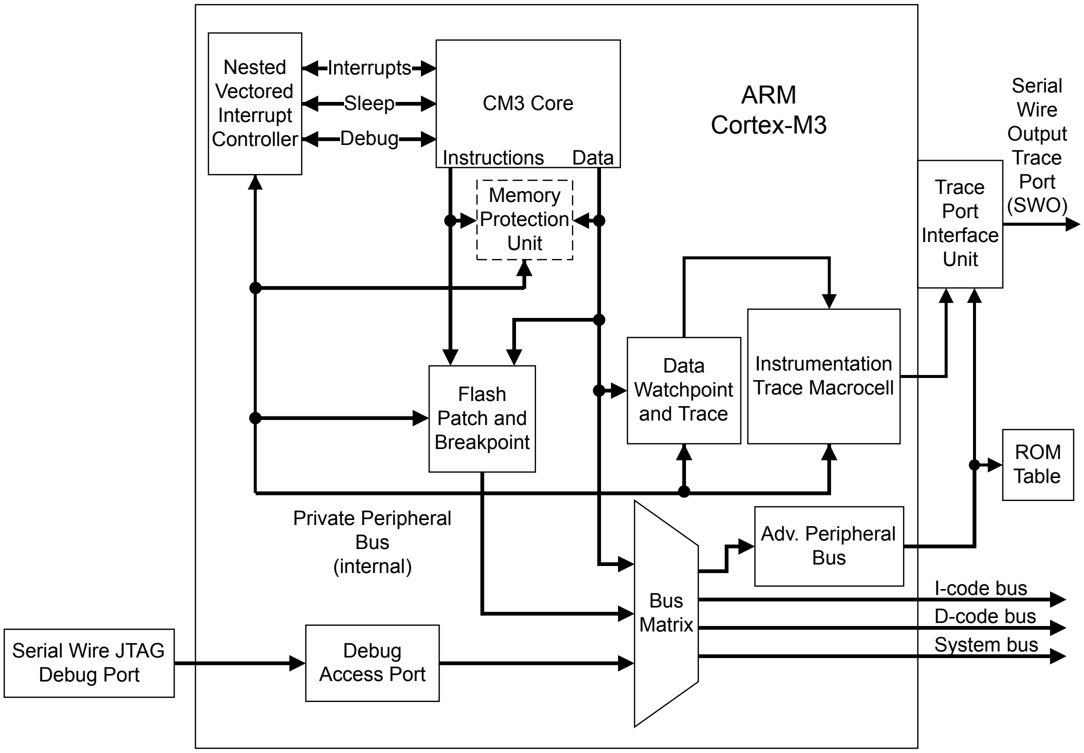
- 32-bit ARM® Cortex™-M3 v7M architecture optimized for small-footprint embedded applications
- Outstanding processing performance combined with fast interrupt handling
- Thumb-2 mixed 16-/32-bit instruction set, delivers the high performance expected of a 32-bit ARM core in a compact memory size usually associated with 8- and 16-bit devices; typically in the range of a few kilobytes of memory for microcontroller-class applications
  - Single-cycle multiply instruction and hardware divide
  - Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
  - Unaligned data access, enabling data to be efficiently packed into memory
- Fast code execution permits slower processor clock or increases sleep mode time
- Harvard architecture characterized by separate buses for instruction and data
- Efficient processor core, system and memories
- Hardware division and fast multiplier
- Deterministic, high-performance interrupt handling for time-critical applications
- Memory protection unit (MPU) to provide a privileged mode for protected operating system functionality
- Enhanced system debug with extensive breakpoint and trace capabilities
- Serial Wire Debug and Serial Wire Trace reduce the number of pins required for debugging and tracing
- Migration from the ARM7™ processor family for better performance and power efficiency
- Optimized for single-cycle Flash memory usage
- Ultra-low power consumption with integrated sleep mode
- 50-MHz operation
- 1.25 DMIPS/MHz

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

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

2.1 Block Diagram

Figure 2-1. CPU Block Diagram



2.2 Functional Description

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

Texas Instruments implements the ARM Cortex-M3 core as shown in Figure 2-1 on page 48. The Cortex-M3 uses the entire 16-bit Thumb instruction set and the base Thumb-2 32-bit instruction set. In addition, as noted in the *ARM® Cortex™-M3 Technical Reference Manual*, several Cortex-M3 components are flexible in their implementation: SW/JTAG-DP, ETM, TPIU, the ROM table, the MPU, and the Nested Vectored Interrupt Controller (NVIC). Each of these is addressed in the sections that follow.

2.2.1 Programming Model

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

- Privileged access and user access - Code can execute as privileged or unprivileged. Unprivileged execution limits or excludes access to some resources. Privileged execution has access to all resources. Handler mode is always privileged. Thread mode can be privileged or unprivileged.

Thread mode is privileged out of reset, but you can change it to user or unprivileged by setting the CONTROL[0] bit using the MSR instruction. User access prevents:

  - Use of some instructions such as CPS to set FAULTMASK and PRIMASK
  - Access to most registers in System Control Space (SCS)

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

- Register set - The processor has the following 32-bit registers:
  - 13 general-purpose registers, r0-r12
  - Stack point alias of banked registers, SP_process and SP_main
  - Link register, r14
  - Program counter, r15
  - One program status register, xPSR.
- Data types - The processor supports the following data types:
  - 32-bit words
  - 16-bit halfwords
  - 8-bit bytes
- Memory formats - The processor views memory as a linear collection of bytes numbered in ascending order from 0. For example, bytes 0-3 hold the first stored word and bytes 4-7 hold the second stored word. The processor accesses code and data in little-endian format, which means that the byte with the lowest address in a word is the least-significant byte of the word. The byte with the highest address in a word is the most significant. The byte at address 0 of the memory system connects to data lines 7-0.
- Instruction set - The Cortex-M3 instruction set contains both 16 and 32-bit instructions. These instructions are summarized in Table 2-1 on page 49 and Table 2-2 on page 51, respectively.

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

Operation	Assembler
Add register value and C flag to register value	ADC <Rd>, <Rm>
Add immediate 3-bit value to register	ADD <Rd>, <Rn>, #<immed_3>
Add immediate 8-bit value to register	ADD <Rd>, #<immed_8>
Add low register value to low register value	ADD <Rd>, <Rn>, <Rm>
Add high register value to low or high register value	ADD <Rd>, <Rm>
Add 4* (immediate 8-bit value) with PC to register	ADD <Rd>, PC, #<immed_8> * 4
Add 4* (immediate 8-bit value) with SP to register	ADD <Rd>, SP, #<immed_8> * 4
Add 4* (immediate 7-bit value) to SP	ADD SP, #<immed_7> * 4
Bitwise AND register values	AND <Rd>, <Rm>
Arithmetic shift right by immediate number	ASR <Rd>, <Rm>, #<immed_5>

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

Operation	Assembler
Arithmetic shift right by number in register	ASR <Rd>, <Rs>
Branch conditional	B<cond> <target address>
Branch unconditional	B <target_address>
Bit clear	BIC <Rd>, <Rm>
Software breakpoint	BKPT <immed_8>
Branch with link	BL <Rm>
Branch with link and exchange	BLX <Rm>
Branch and exchange	BX <Rm>
Compare not zero and branch	CBNZ <Rn>,<label>
Compare zero and branch	CBZ <Rn>,<label>
Compare negation of register value with another register value	CMN <Rn>, <Rm>
Compare immediate 8-bit value	CMP <Rn>, #<immed_8>
Compare registers	CMP <Rn>, <Rm>
Compare high register to low or high register	CMP <Rn>, <Rm>
Change processor state	CPS <effect>, <iflags>
Copy high or low register value to another high or low register	CPY <Rd> <Rm>
Bitwise exclusive OR register values	EOR <Rd>, <Rm>
Condition the following instruction	IT <cond>
Condition the following two instructions	IT<x> <cond>
Condition the following three instructions	IT<x><y> <cond>
Condition the following four instructions	IT<x><y><z> <cond>
Multiple sequential memory word loads	LDMIA <Rn>!, <registers>
Load memory word from base register address + 5-bit immediate offset	LDR <Rd>, [<Rn>, #<immed_5> * 4]
Load memory word from base register address + register offset	LDR <Rd>, [<Rn>, <Rm>]
Load memory word from PC address + 8-bit immediate offset	LDR <Rd>, [PC, #<immed_8> * 4]
Load memory word from SP address + 8-bit immediate offset	LDR, <Rd>, [SP, #<immed_8> * 4]
Load memory byte [7:0] from register address + 5-bit immediate offset	LDRB <Rd>, [<Rn>, #<immed_5>]
Load memory byte [7:0] from register address + register offset	LDRB <Rd>, [<Rn>, <Rm>]
Load memory halfword [15:0] from register address + 5-bit immediate offset	LDRH <Rd>, [<Rn>, #<immed_5> * 2]
Load halfword [15:0] from register address + register offset	LDRH <Rd>, [<Rn>, <Rm>]
Load signed byte [7:0] from register address + register offset	LDRSB <Rd>, [<Rn>, <Rm>]
Load signed halfword [15:0] from register address + register offset	LDRSH <Rd>, [<Rn>, <Rm>]
Logical shift left by immediate number	LSL <Rd>, <Rm>, #<immed_5>
Logical shift left by number in register	LSL <Rd>, <Rs>
Logical shift right by immediate number	LSR <Rd>, <Rm>, #<immed_5>
Logical shift right by number in register	LSR <Rd>, <Rs>
Move immediate 8-bit value to register	MOV <Rd>, #<immed_8>
Move low register value to low register	MOV <Rd>, <Rn>
Move high or low register value to high or low register	MOV <Rd>, <Rm>
Multiply register values	MUL <Rd>, <Rm>
Move complement of register value to register	MVN <Rd>, <Rm>
Negate register value and store in register	NEG <Rd>, <Rm>

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

Operation	Assembler
No operation	NOP <c>
Bitwise logical OR register values	ORR <Rd>, <Rm>
Pop registers from stack	POP <registers>
Pop registers and PC from stack	POP <registers, PC>
Push registers onto stack	PUSH <registers>
Push LR and registers onto stack	PUSH <registers, LR>
Reverse bytes in word and copy to register	REV <Rd>, <Rn>
Reverse bytes in two halfwords and copy to register	REV16 <Rd>, <Rn>
Reverse bytes in low halfword [15:0], sign-extend, and copy to register	REVSH <Rd>, <Rn>
Rotate right by amount in register	ROR <Rd>, <Rs>
Subtract register value and C flag from register value	SBC <Rd>, <Rm>
Send event	SEV <c>
Store multiple register words to sequential memory locations	STMIA <Rn>!, <registers>
Store register word to register address + 5-bit immediate offset	STR <Rd>, [<Rn>, #<immed_5> * 4]
Store register word to register address	STR <Rd>, [<Rn>, <Rm>]
Store register word to SP address + 8-bit immediate offset	STR <Rd>, [SP, #<immed_8> * 4]
Store register byte [7:0] to register address + 5-bit immediate offset	STRB <Rd>, [<Rn>, #<immed_5>]
Store register byte [7:0] to register address	STRB <Rd>, [<Rn>, <Rm>]
Store register halfword [15:0] to register address + 5-bit immediate offset	STRH <Rd>, [<Rn>, #<immed_5> * 2]
Store register halfword [15:0] to register address + register offset	STRH <Rd>, [<Rn>, <Rm>]
Subtract immediate 3-bit value from register	SUB <Rd>, <Rn>, #<immed_3>
Subtract immediate 8-bit value from register value	SUB <Rd>, #<immed_8>
Subtract register values	SUB <Rd>, <Rn>, <Rm>
Subtract 4 (immediate 7-bit value) from SP	SUB SP, #<immed_7> * 4
Operating system service call with 8-bit immediate call code	SVC <immed_8>
Extract byte [7:0] from register, move to register, and sign-extend to 32 bits	SXTB <Rd>, <Rm>
Extract halfword [15:0] from register, move to register, and sign-extend to 32 bits	SXTH <Rd>, <Rm>
Test register value for set bits by ANDing it with another register value	TST <Rn>, <Rm>
Extract byte [7:0] from register, move to register, and zero-extend to 32 bits	UXTB <Rd>, <Rm>10
Extract halfword [15:0] from register, move to register, and zero-extend to 32 bits	UXTH <Rd>, <Rm>
Wait for event	WFE <c>
Wait for interrupt	WFI <c>

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

Operation	Assembler
Add register value, immediate 12-bit value, and C bit	ADC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12>
Add register value, shifted register value, and C bit	ADC{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
Add register value and immediate 12-bit value	ADD{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
Add register value and shifted register value	ADD{S}.W <Rd>, <Rm>{, <shift>}
Add register value and immediate 12-bit value	ADDW.W <Rd>, <Rn>, #<immed_12>
Bitwise AND register value with immediate 12-bit value	AND{S}.W <Rd>, <Rn>, #<modify_constant(immed_12>

Table 2-2. 32-Bit Cortex-M3 Instruction Set Summary *(continued)*

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

Table 2-2. 32-Bit Cortex-M3 Instruction Set Summary *(continued)*

Operation	Assembler
Memory doubleword from register address 8-bit offset 4, postindexed	LDRD.W <Rxf>, <Rxf2>, [<Rn>], #+/–<offset_8> * 4
Load register exclusive calculates an address from a base register value and an immediate offset, loads a word from memory, writes it to a register	LDREX<c> <Rt>,[<Rn>{,#<imm>}]
Load register exclusive halfword calculates an address from a base register value and an immediate offset, loads a halfword from memory, writes it to a register	LDREXH<c> <Rt>,[<Rn>{,#<imm>}]
Load register exclusive byte calculates an address from a base register value and an immediate offset, loads a byte from memory, writes it to a register	LDREXB<c> <Rt>,[<Rn>{,#<imm>}]
Memory halfword [15:0] from base register address + immediate 12-bit offset	LDRH.W <Rxf>, [<Rn>, #<offset_12>]
Memory halfword [15:0] from base register address immediate 8-bit offset, preindexed	LDRH.W <Rxf>, [<Rn>, #<+/–<offset_8>]!
Memory halfword [15:0] from base register address immediate 8-bit offset, postindexed	LDRH.W <Rxf>. [<Rn>], #+/-<offset_8>
Memory halfword [15:0] from register address shifted left by 0, 1, 2, or 3 places	LDRH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
Memory halfword from PC address immediate 12-bit offset	LDRH.W <Rxf>, [PC, #+/–<offset_12>]
Memory signed byte [7:0] from base register address + immediate 12-bit offset	LDRSB.W <Rxf>, [<Rn>, #<offset_12>]
Memory signed byte [7:0] from base register address immediate 8-bit offset, postindexed	LDRSB.W <Rxf>. [<Rn>], #+/-<offset_8>
Memory signed byte [7:0] from base register address immediate 8-bit offset, preindexed	LDRSB.W <Rxf>, [<Rn>, #<+/–<offset_8>]!
Memory signed byte [7:0] from register address shifted left by 0, 1, 2, or 3 places	LDRSB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
Memory signed byte from PC address immediate 12-bit offset	LDRSB.W <Rxf>, [PC, #+/–<offset_12>]
Memory signed halfword [15:0] from base register address + immediate 12-bit offset	LDRSH.W <Rxf>, [<Rn>, #<offset_12>]
Memory signed halfword [15:0] from base register address immediate 8-bit offset, postindexed	LDRSH.W <Rxf>. [<Rn>], #+/-<offset_8>
Memory signed halfword [15:0] from base register address immediate 8-bit offset, preindexed	LDRSH.W <Rxf>, [<Rn>, #<+/–<offset_8>]!
Memory signed halfword [15:0] from register address shifted left by 0, 1, 2, or 3 places	LDRSH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
Memory signed halfword from PC address immediate 12-bit offset	LDRSH.W <Rxf>, [PC, #+/–<offset_12>]
Logical shift left register value by number in register	LSL{S}.W <Rd>, <Rn>, <Rm>
Logical shift right register value by number in register	LSR{S}.W <Rd>, <Rn>, <Rm>
Multiply two signed or unsigned register values and add the low 32 bits to a register value	MLA.W <Rd>, <Rn>, <Rm>, <Racc>
Multiply two signed or unsigned register values and subtract the low 32 bits from a register value	MLS.W <Rd>, <Rn>, <Rm>, <Racc>
Move immediate 12-bit value to register	MOV{S}.W <Rd>, #<modify_constant(immed_12)>
Move shifted register value to register	MOV{S}.W <Rd>, <Rm>{, <shift>}
Move immediate 16-bit value to top halfword [31:16] of register	MOVT.W <Rd>, #<immed_16>
Move immediate 16-bit value to bottom halfword [15:0] of register and clear top halfword [31:16]	MOVW.W <Rd>, #<immed_16>
Move to register from status	MRS<c> <Rd>, <psr>
Move to status register	MSR<c> <psr>_<fields>,<Rn>
Multiply two signed or unsigned register values	MUL.W <Rd>, <Rn>, <Rm>
No operation	NOP.W

Table 2-2. 32-Bit Cortex-M3 Instruction Set Summary *(continued)*

Operation	Assembler
Logical OR NOT register value with immediate 12-bit value	ORN{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
Logical OR NOT register value with shifted register value	ORN[S}.W <Rd>, <Rn>, <Rm>{, <shift>}
Logical OR register value with immediate 12-bit value	ORR{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
Logical OR register value with shifted register value	ORR{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
Reverse bit order	RBIT.W <Rd>, <Rm>
Reverse bytes in word	REV.W <Rd>, <Rm>
Reverse bytes in each halfword	REV16.W <Rd>, <Rn>
Reverse bytes in bottom halfword and sign-extend	REVSH.W <Rd>, <Rn>
Rotate right by number in register	ROR{S}.W <Rd>, <Rn>, <Rm>
Rotate right with extend	RRX{S}.W <Rd>, <Rm>
Subtract a register value from an immediate 12-bit value	RSB{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
Subtract a register value from a shifted register value	RSB{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
Subtract immediate 12-bit value and C bit from register value	SBC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
Subtract shifted register value and C bit from register value	SBC{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
Copy selected bits to register and sign-extend	SBFX.W <Rd>, <Rn>, #<lsb>, #<width>
Signed divide	SDIV<c> <Rd>,<Rn>,<Rm>
Send event	SEV<c>
Multiply signed words and add signed-extended value to 2-register value	SMLAL.W <RdLo>, <RdHi>, <Rn>, <Rm>
Multiply two signed register values	SMULL.W <RdLo>, <RdHi>, <Rn>, <Rm>
Signed saturate	SSAT.W <c> <Rd>, #<imm>, <Rn>{, <shift>}
Multiple register words to consecutive memory locations	STM{IA|DB}.W <Rn>{!}, <registers>
Register word to register address + immediate 12-bit offset	STR.W <Rxf>, [<Rn>, #<offset_12>]
Register word to register address immediate 8-bit offset, postindexed	STR.W <Rxf>, [<Rn>], #+/–<offset_8>
Register word to register address shifted by 0, 1, 2, or 3 places	STR.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
Register word to register address immediate 8-bit offset, preindexed Store, preindexed	STR.W <Rxf>, [<Rn>, #+/-<offset_8>]{!} STRT.W <Rxf>, [<Rn>, #<offset_8>]
Register byte [7:0] to register address immediate 8-bit offset, preindexed	STRB{T}.W <Rxf>, [<Rn>, #+/–<offset_8>]{!}
Register byte [7:0] to register address + immediate 12-bit offset	STRB.W <Rxf>, [<Rn>, #<offset_12>]
Register byte [7:0] to register address immediate 8-bit offset, postindexed	STRB.W <Rxf>, [<Rn>], #+/–<offset_8>
Register byte [7:0] to register address shifted by 0, 1, 2, or 3 places	STRB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
Store doubleword, preindexed	STRD.W <Rxf>, <Rxf2>, [<Rn>, #+/–<offset_8> * 4]{!}
Store doubleword, postindexed	STRD.W <Rxf>, <Rxf2>, [<Rn>, #+/–<offset_8> * 4]
Store register exclusive calculates an address from a base register value and an immediate offset, and stores a word from a register to memory if the executing processor has exclusive access to the memory addressed.	STREX <c> <Rd>,<Rt>,[<Rn>{,#<imm>}]
Store register exclusive byte derives an address from a base register value, and stores a byte from a register to memory if the executing processor has exclusive access to the memory addressed	STREXB <c> <Rd>,<Rt>,[<Rn>]
Store register exclusive halfword derives an address from a base register value, and stores a halfword from a register to memory if the executing processor has exclusive access to the memory addressed.	STREXH <c> <Rd>,<Rt>,[<Rn>]
Register halfword [15:0] to register address + immediate 12-bit offset	STRH.W <Rxf>, [<Rn>, #<offset_12>]
Register halfword [15:0] to register address shifted by 0, 1, 2, or 3 places	STRH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]
Register halfword [15:0] to register address immediate 8-bit offset, preindexed	STRH{T}.W <Rxf>, [<Rn>, #+/–<offset_8>]{!}

Table 2-2. 32-Bit Cortex-M3 Instruction Set Summary *(continued)*

Operation	Assembler
Register halfword [15:0] to register address immediate 8-bit offset, postindexed	STRH.W <Rxf>, [<Rn>], #+/–<offset_8>
Subtract immediate 12-bit value from register value	SUB{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>
Subtract shifted register value from register value	SUB{S}.W <Rd>, <Rn>, <Rm>{, <shift>}
Subtract immediate 12-bit value from register value	SUBW.W <Rd>, <Rn>, #<immed_12>
Sign extend byte to 32 bits	SXTB.W <Rd>, <Rm>{, <rotation>}
Sign extend halfword to 32 bits	SXTH.W <Rd>, <Rm>{, <rotation>}
Table branch byte	TBB [<Rn>, <Rm>]
Table branch halfword	TBH [<Rn>, <Rm>, LSL #1]
Exclusive OR register value with immediate 12-bit value	TEQ.W <Rn>, #<modify_constant(immed_12)>
Exclusive OR register value with shifted register value	TEQ.W <Rn>, <Rm>{, <shift}
Logical AND register value with 12-bit immediate value	TST.W <Rn>, #<modify_constant(immed_12)>
Logical AND register value with shifted register value	TST.W <Rn>, <Rm>{, <shift>}
Copy bit field from register value to register and zero-extend to 32 bits	UBFX.W <Rd>, <Rn>, #<lsb>, #<width>
Unsigned divide	UDIV<c> <Rd>,<Rn>,<Rm>
Multiply two unsigned register values and add to a 2-register value	UMLAL.W <RdLo>, <RdHi>, <Rn>, <Rm>
Multiply two unsigned register values	UMULL.W <RdLo>, <RdHi>, <Rn>, <Rm>
Unsigned saturate	USAT <c> <Rd>, #<imm>, <Rn>{, <shift>}
Copy unsigned byte to register and zero-extend to 32 bits	UXTB.W <Rd>, <Rm>{, <rotation>}
Copy unsigned halfword to register and zero-extend to 32 bits	UXTH.W <Rd>, <Rm>{, <rotation>}
Wait for event	WFE.W
Wait for interrupt	WFI.W

2.2.2 Serial Wire and JTAG Debug

Texas Instruments replaces the ARM SW-DP and JTAG-DP with the ARM CoreSight™-compliant Serial Wire JTAG Debug Port (SWJ-DP) interface. As a result, Chapter 12, “Debug Port,” of the *ARM® Cortex™-M3 Technical Reference Manual* does not apply to Stellaris® devices.

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

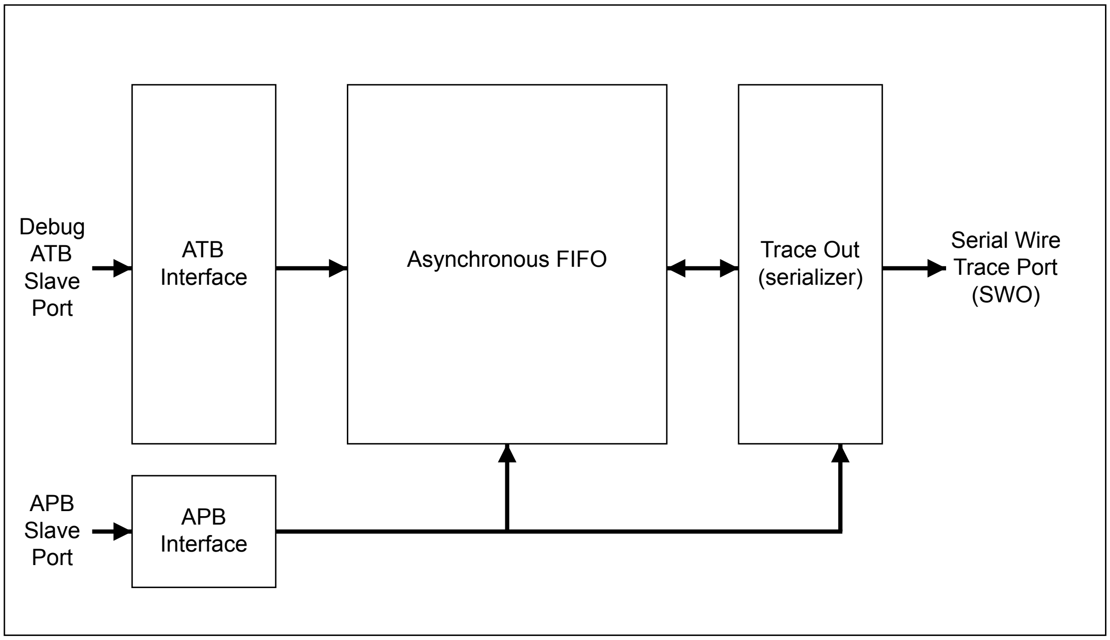
2.2.3 Embedded Trace Macrocell (ETM)

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

2.2.4 Trace Port Interface Unit (TPIU)

The TPIU acts as a bridge between the Cortex-M3 trace data from the ITM, and an off-chip Trace Port Analyzer. Stellaris® devices implement the TPIU as shown in Figure 2-2. This implementation is similar to the non-ETM version described in the *ARM® Cortex™-M3 Technical Reference Manual*, however, SWJ-DP only provides the Serial Wire Viewer (SWV) output format for the TPIU.

Figure 2-2. TPIU Block Diagram



2.2.5 ROM Table

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

2.2.6 Memory Protection Unit (MPU)

The Memory Protection Unit (MPU) is included on the LM3S1W16 controller and supports the standard ARMv7 Protected Memory System Architecture (PMSA) model. The MPU provides full support for protection regions, overlapping protection regions, access permissions, and exporting memory attributes to the system.

2.2.7 Nested Vectored Interrupt Controller (NVIC)

The Nested Vectored Interrupt Controller (NVIC):

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

The NVIC and the processor core interface are closely coupled, which enables low latency interrupt processing and efficient processing of late arriving interrupts. The NVIC maintains knowledge of the stacked (nested) interrupts to enable tail-chaining of interrupts.

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

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

2.2.7.1 Interrupts

The *ARM® Cortex™-M3 Technical Reference Manual* describes the maximum number of interrupts and interrupt priorities. The LM3S1W16 microcontroller supports 30 interrupts with eight priority levels.

In addition to the peripheral interrupts, the system also provides for a non-maskable interrupt (NMI). The NMI is generally used in safety critical applications where the immediate execution of an interrupt handler is required. The `NMI` signal is available as an external signal so that it may be generated by external circuitry. The NMI is also used internally as part of the main oscillator verification circuitry. More information on the non-maskable interrupt is located in “Non-Maskable Interrupt” on page 82.

2.2.8 System Timer (SysTick)

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

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

2.2.8.1 Functional Description

The timer consists of three registers:

- SysTick Control and Status Register - a control and status counter to configure its clock, enable the counter, enable the SysTick interrupt, and determine counter status
- SysTick Reload Value Register - the reload value for the counter, used to provide the counter's wrap value
- SysTick Current Value Register - the current value of the counter

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

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

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

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

2.2.8.2 SysTick Control and Status Register

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

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	COUNTFLAG	R/W	0	Count Flag When set, this bit indicates that the timer has counted to 0 since the last time this register was read. This bit is cleared by a read of the register. If read by the debugger using the DAP, this bit is cleared only if the MasterType bit in the AHB-AP Control Register is clear. Otherwise, the COUNTFLAG bit is not changed by the debugger read.
15:3	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	CLKSOURCE	R/W	0	Clock Source Value Description 0 External reference clock. (Not implemented for Stellaris® microcontrollers.) 1 Core clock Because an external reference clock is not supported, this bit must be set in order for SysTick to operate.
1	TICKINT	R/W	0	Tick Interrupt When set, this bit causes an interrupt to be generated to the NVIC when SysTick counts to 0. When clear, interrupt generation is disabled. Software can use the COUNTFLAG to determine if the counter has ever reached 0.
0	ENABLE	R/W	0	Enable When set, this bit enables SysTick to operate in a multi-shot way. That is, the counter loads the Reload value and begins counting down. On reaching 0, the COUNTFLAG bit is set and an interrupt is generated if enabled by TICKINT. The counter then loads the Reload value again and begins counting. When this bit is clear, the counter is disabled.

2.2.8.3 SysTick Reload Value Register

The SysTick Reload Value Register specifies the start value to load into the SysTick Current Value Register when the counter reaches 0. The start value can be between 1 and 0x00FF.FFFF. A start value of 0 is possible but has no effect because the SysTick interrupt and COUNTFLAG are activated when counting from 1 to 0.

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

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

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

2.2.8.4 SysTick Current Value Register

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

Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:0	CURRENT	W1C	-	Current Value This field contains the current value at the time the register is accessed. No read-modify-write protection is provided, so change with care. This register is write-clear. Writing to it with any value clears the register to 0. Clearing this register also clears the COUNTFLAG bit of the SysTick Control and Status Register.

2.2.8.5 SysTick Calibration Value Register

The SysTick Calibration Value register is not implemented.

3 Memory Map

The memory map for the LM3S1W16 controller is provided in Table 3-1.

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

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

Table 3-1. Memory Map

Start	End	Description	For details, see page ...
Memory			
0x0000.0000	0x0000.7FFF	On-chip Flash	195
0x0000.8000	0x00FF.FFFF	Reserved	-
0x0100.0000	0x1FFF.FFFF	Reserved for ROM	195
0x2000.0000	0x2000.1FFF	Bit-banded on-chip SRAM	195
0x2000.2000	0x21FF.FFFF	Reserved	-
0x2200.0000	0x2203.FFFF	Bit-band alias of 0x2000.0000 through 0x200F.FFFF	195
0x2204.0000	0x3FFF.FFFF	Reserved	-
FiRM Peripherals			
0x4000.0000	0x4000.0FFF	Watchdog timer 0	384
0x4000.1000	0x4000.1FFF	Watchdog timer 1	384
0x4000.2000	0x4000.3FFF	Reserved	-
0x4000.4000	0x4000.4FFF	GPIO Port A	296
0x4000.5000	0x4000.5FFF	GPIO Port B	296
0x4000.6000	0x4000.6FFF	GPIO Port C	296
0x4000.7000	0x4000.7FFF	GPIO Port D	296
0x4000.8000	0x4000.8FFF	SSI0	549
0x4000.9000	0x4000.9FFF	SSI1	549
0x4000.A000	0x4000.BFFF	Reserved	-
0x4000.C000	0x4000.CFFF	UART0	489
0x4000.D000	0x4000.DFFF	UART1	489
0x4000.E000	0x4000.EFFF	UART2	489
0x4000.F000	0x4001.FFFF	Reserved	-
Peripherals			
0x4002.0000	0x4002.07FF	I^2C Master 0	592
0x4002.0800	0x4002.0FFF	I^2C Slave 0	605
0x4002.1000	0x4002.17FF	I^2C Master 1	592
0x4002.1800	0x4002.1FFF	I^2C Slave 1	605
0x4002.2000	0x4002.3FFF	Reserved	-
0x4002.4000	0x4002.4FFF	GPIO Port E	296
0x4002.5000	0x4002.FFFF	Reserved	-
0x4003.0000	0x4003.0FFF	Timer 0	352
0x4003.1000	0x4003.1FFF	Timer 1	352

Table 3-1. Memory Map *(continued)*

Start	End	Description	For details, see page ...
0x4003.2000	0x4003.2FFF	Timer 2	352
0x4003.3000	0x4003.7FFF	Reserved	-
0x4003.8000	0x4003.8FFF	ADC0	423
0x4003.9000	0x4003.BFFF	Reserved	-
0x4003.C000	0x4003.CFFF	Analog Comparators	614
0x4003.D000	0x4005.7FFF	Reserved	-
0x4005.8000	0x4005.8FFF	GPIO Port A (AHB aperture)	296
0x4005.9000	0x4005.9FFF	GPIO Port B (AHB aperture)	296
0x4005.A000	0x4005.AFFF	GPIO Port C (AHB aperture)	296
0x4005.B000	0x4005.BFFF	GPIO Port D (AHB aperture)	296
0x4005.C000	0x4005.CFFF	GPIO Port E (AHB aperture)	296
0x4005.D000	0x400F.BFFF	Reserved	-
0x400F.C000	0x400F.CFFF	Hibernation Module	177
0x400F.D000	0x400F.DFFF	Flash memory control	200
0x400F.E000	0x400F.EFFF	System control	93
0x400F.F000	0x400F.FFFF	μDMA	248
0x4010.0000	0x41FF.FFFF	Reserved	-
0x4200.0000	0x43FF.FFFF	Bit-banded alias of 0x4000.0000 through 0x400F.FFFF	-
0x4400.0000	0xDFFF.FFFF	Reserved	-
Private Peripheral Bus			
0xE000.0000	0xE000.0FFF	Instrumentation Trace Macrocell (ITM)	*ARM® Cortex™-M3 Technical Reference Manual*
0xE000.1000	0xE000.1FFF	Data Watchpoint and Trace (DWT)	*ARM® Cortex™-M3 Technical Reference Manual*
0xE000.2000	0xE000.2FFF	Flash Patch and Breakpoint (FPB)	*ARM® Cortex™-M3 Technical Reference Manual*
0xE000.3000	0xE000.DFFF	Reserved	-
0xE000.E000	0xE000.EFFF	Nested Vectored Interrupt Controller (NVIC)	*ARM® Cortex™-M3 Technical Reference Manual*
0xE000.F000	0xE003.FFFF	Reserved	-
0xE004.0000	0xE004.0FFF	Trace Port Interface Unit (TPIU)	*ARM® Cortex™-M3 Technical Reference Manual*

Table 3-1. Memory Map *(continued)*

Start	End	Description	For details, see page ...
0xE004.1000	0xFFFF.FFFF	Reserved	-

4 Interrupts

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

Table 4-1 on page 63 lists all exception types. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 30 interrupts (listed in Table 4-2 on page 64).

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

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

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

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

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

Table 4-1. Exception Types

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

Table 4-1. Exception Types *(continued)*

Exception Type	Vector Number	Priority[a]	Description
Bus Fault	5	programmable	This exception is caused by a pre-fetch fault, memory access fault, and other address/memory related faults. This exception is synchronous when precise and asynchronous when imprecise. This fault can be enabled or disabled.
Usage Fault	6	programmable	This exception is caused by a usage fault, such as undefined instruction executed or illegal state transition attempt. This exception is synchronous.
-	7-10	-	Reserved.
SVCall	11	programmable	This exception is caused by a system service call with an SVC instruction. This exception is synchronous.
Debug Monitor	12	programmable	This exception is caused by the debug monitor (when not halting). This exception is synchronous, but only active when enabled. This exception does not activate if it is a lower priority than the current activation.
-	13	-	Reserved.
PendSV	14	programmable	This exception is caused by a pendable request for system service. This exception is asynchronous and only pended by software.
SysTick	15	programmable	This exception is caused by the SysTick timer reaching 0, when it is enabled to generate an interrupt. This exception is asynchronous.
Interrupts	16 and above	programmable	This exception is caused by interrupts asserted from outside the ARM Cortex-M3 core and fed through the NVIC (prioritized). These exceptions are all asynchronous. Table 4-2 on page 64 lists the interrupts on the LM3S1W16 controller.

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

Table 4-2. Interrupts

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Description
0-15	-	Processor exceptions
16	0	GPIO Port A
17	1	GPIO Port B
18	2	GPIO Port C
19	3	GPIO Port D
20	4	GPIO Port E
21	5	UART0
22	6	UART1
23	7	SSI0
24	8	I^{2}C0
25-29	9-13	Reserved
30	14	ADC0 Sequence 0
31	15	ADC0 Sequence 1
32	16	ADC0 Sequence 2
33	17	ADC0 Sequence 3
34	18	Watchdog Timers 0 and 1
35	19	Timer 0A
36	20	Timer 0B

Table 4-2. Interrupts *(continued)*

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Description
37	21	Timer 1A
38	22	Timer 1B
39	23	Timer 2A
40	24	Timer 2B
41	25	Analog Comparator 0
42	26	Analog Comparator 1
43	27	Reserved
44	28	System Control
45	29	Flash Memory Control
46-48	30-32	Reserved
49	33	UART2
50	34	SSI1
51-52	35-36	Reserved
53	37	I^2C1
54-58	38-42	Reserved
59	43	Hibernation Module
60-61	44-45	Reserved
62	46	μDMA Software
63	47	μDMA Error
64-71	48-55	Reserved

5 JTAG Interface

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

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

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

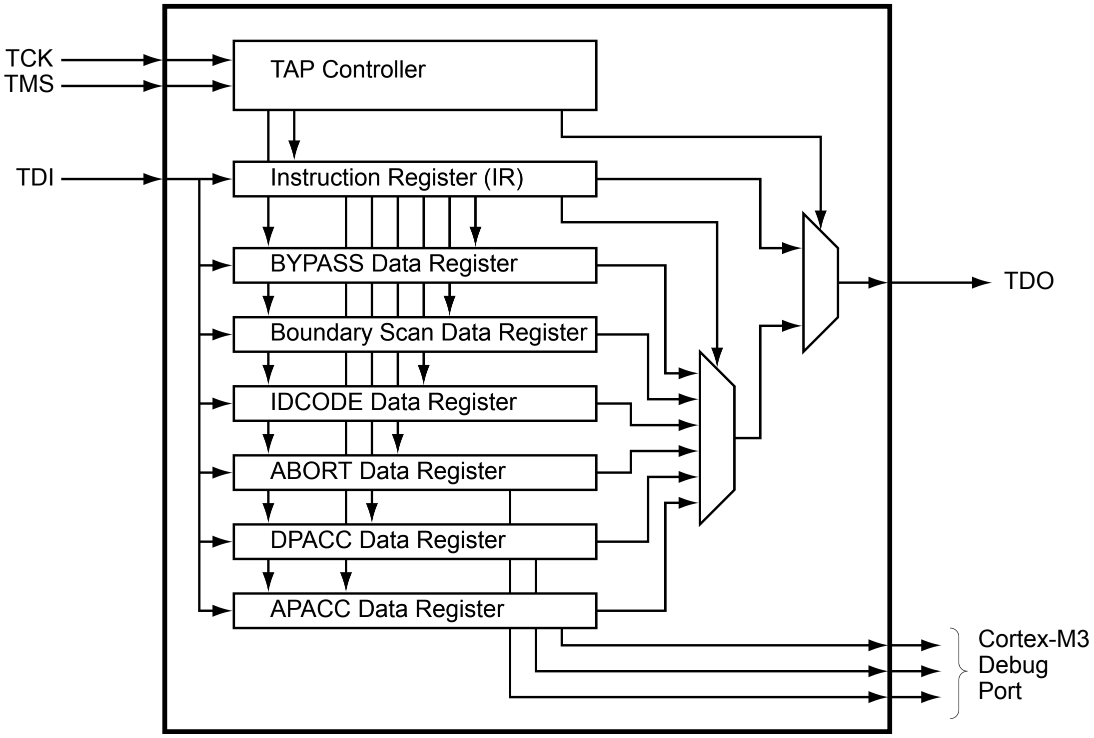
The Stellaris® JTAG module has the following features:

- IEEE 1149.1-1990 compatible Test Access Port (TAP) controller
- Four-bit Instruction Register (IR) chain for storing JTAG instructions
- IEEE standard instructions: BYPASS, IDCODE, SAMPLE/PRELOAD, EXTEST and INTEST
- ARM additional instructions: APACC, DPACC and ABORT
- Integrated ARM Serial Wire Debug (SWD)
  - Serial Wire JTAG Debug Port (SWJ-DP)
  - Flash Patch and Breakpoint (FPB) unit for implementing breakpoints
  - Data Watchpoint and Trigger (DWT) unit for implementing watchpoints, trigger resources, and system profiling
  - Instrumentation Trace Macrocell (ITM) for support of printf style debugging
  - Trace Port Interface Unit (TPIU) for bridging to a Trace Port Analyzer

See the *ARM® 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 Signal Description

Table 5-1 on page 67 lists the external signals of the JTAG/SWD controller and describes the function of each. The JTAG/SWD controller signals are alternate functions for some GPIO signals, however note that the reset state of the pins is for the JTAG/SWD function. The JTAG/SWD controller signals are under commit protection and require a special process to be configured as GPIOs, see "Commit Control" on page 291. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the JTAG/SWD controller signals. The `AFSEL` bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 307) is set to choose the JTAG/SWD function.The number in parentheses is the encoding that must be programmed into the `PMCn` field in the **GPIO Port Control (GPIOPCTL)** register (page 324) to assign the JTAG/SWD controller signals to the specified GPIO port pin. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 285.

Table 5-1. Signals for JTAG_SWD_SWO

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`SWCLK`	52	PC0 (3)	I	TTL	JTAG/SWD CLK.
`SWDIO`	51	PC1 (3)	I/O	TTL	JTAG TMS and SWDIO.
`SWO`	49	PC3 (3)	O	TTL	JTAG TDO and SWO.
`TCK`	52	PC0 (3)	I	TTL	JTAG/SWD CLK.
`TDI`	50	PC2 (3)	I	TTL	JTAG TDI.
`TDO`	49	PC3 (3)	O	TTL	JTAG TDO and SWO.

Table 5-1. Signals for JTAG_SWD_SWO *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`TMS`	51	PC1 (3)	I	TTL	JTAG TMS and SWDIO.

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

5.3 Functional Description

A high-level conceptual drawing of the JTAG module is shown in Figure 5-1 on page 67. The JTAG module is composed of the Test Access Port (TAP) controller and serial shift chains with parallel update registers. The TAP controller is a simple state machine controlled by the `TCK` and `TMS` inputs. The current state of the TAP controller depends on the sequence of values captured on `TMS` at the rising edge of `TCK`. The TAP controller determines when the serial shift chains capture new data, shift data from `TDI` towards `TDO`, and update the parallel load registers. The current state of the TAP controller also determines whether the Instruction Register (IR) chain or one of the Data Register (DR) chains is being accessed.

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

Some instructions, like EXTEST and INTEST, operate on data currently in a DR chain and do not capture, shift, or update any of the chains. Instructions that are not implemented decode to the BYPASS instruction to ensure that the serial path between `TDI` and `TDO` is always connected (see Table 5-3 on page 74 for a list of implemented instructions).

See “JTAG and Boundary Scan” on page 648 for JTAG timing diagrams.

Note: Of all the possible reset sources, only Power-On reset (POR) and the assertion of the $\overline{\text{RST}}$ input have any effect on the JTAG module. The pin configurations are reset by both the $\overline{\text{RST}}$ input and POR, whereas the internal JTAG logic is only reset with POR. See “Reset Sources” on page 79 for more information on reset.

5.3.1 JTAG Interface Pins

The JTAG interface consists of four standard pins: `TCK`, `TMS`, `TDI`, and `TDO`. These pins and their associated state after a power-on reset or reset caused by the $\overline{\text{RST}}$ input are given in Table 5-2. Detailed information on each pin follows. Refer to “General-Purpose Input/Outputs (GPIOs)” on page 285 for information on how to reprogram the configuration of these pins.

Table 5-2. JTAG Port Pins State after Power-On Reset or $\overline{\text{RST}}$ assertion

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

5.3.1.1 Test Clock Input (TCK)

The `TCK` pin is the clock for the JTAG module. This clock is provided so the test logic can operate independently of any other system clocks and to ensure that multiple JTAG TAP controllers that are daisy-chained together can synchronously communicate serial test data between components.

During normal operation, `TCK` is driven by a free-running clock with a nominal 50% duty cycle. When necessary, `TCK` can be stopped at 0 or 1 for extended periods of time. While `TCK` is stopped at 0 or 1, the state of the TAP controller does not change and data in the JTAG Instruction and Data Registers is not lost.

By default, the internal pull-up resistor on the `TCK` pin is enabled after reset, assuring that no clocking occurs if the pin is not driven from an external source. The internal pull-up and pull-down resistors can be turned off to save internal power as long as the `TCK` pin is constantly being driven by an external source (see page 313 and page 315).

5.3.1.2 Test Mode Select (TMS)

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

Holding `TMS` high for five consecutive `TCK` cycles drives the TAP controller state machine to the Test-Logic-Reset state. When the TAP controller enters the Test-Logic-Reset state, the JTAG module and associated registers are reset to their default values. This procedure should be performed to initialize the JTAG controller. The JTAG Test Access Port state machine can be seen in its entirety in Figure 5-2 on page 70.

By default, the internal pull-up resistor on the `TMS` pin is enabled after reset. Changes to the pull-up resistor settings on GPIO Port C should ensure that the internal pull-up resistor remains enabled on `PC1/TMS`; otherwise JTAG communication could be lost (see page 313).

5.3.1.3 Test Data Input (TDI)

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

By default, the internal pull-up resistor on the `TDI` pin is enabled after reset. Changes to the pull-up resistor settings on GPIO Port C should ensure that the internal pull-up resistor remains enabled on `PC2/TDI`; otherwise JTAG communication could be lost (see page 313).

5.3.1.4 Test Data Output (TDO)

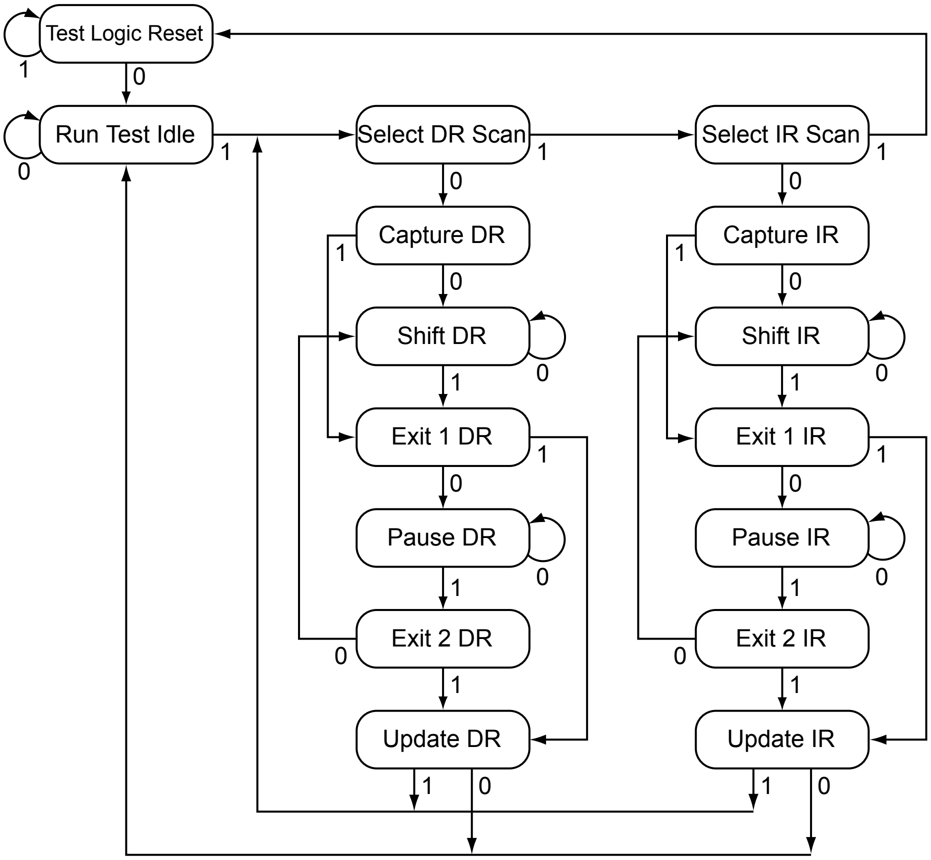
The `TDO` pin provides an output stream of serial information from the IR chain or the DR chains. The value of `TDO` depends on the current TAP state, the current instruction, and the data in the chain being accessed. In order to save power when the JTAG port is not being used, the `TDO` pin is placed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

By default, the internal pull-up resistor on the `TDO` pin is enabled after reset, assuring that the pin remains at a constant logic level when the JTAG port is not being used. The internal pull-up and pull-down resistors can be turned off to save internal power if a High-Z output value is acceptable during certain TAP controller states (see page 313 and page 315).

5.3.2 JTAG TAP Controller

The JTAG TAP controller state machine is shown in Figure 5-2. The TAP controller state machine is reset to the Test-Logic-Reset state on the assertion of a Power-On-Reset (POR). In order to reset the JTAG module after the microcontroller has been powered on, the `TMS` input must be held HIGH for five TCK clock cycles, resetting the TAP controller and all associated JTAG chains. Asserting the correct sequence on the `TMS` pin allows the JTAG module to shift in new instructions, shift in data, or idle during extended testing sequences. For detailed information on the function of the TAP controller and the operations that occur in each state, please refer to *IEEE Standard 1149.1*.

Figure 5-2. Test Access Port State Machine



5.3.3 Shift Registers

The Shift Registers consist of a serial shift register chain and a parallel load register. The serial shift register chain samples specific information during the TAP controller's CAPTURE states and allows this information to be shifted out on `TDO` during the TAP controller's SHIFT states. While the sampled data is being shifted out of the chain on `TDO`, new data is being shifted into the serial shift register on `TDI`. This new data is stored in the parallel load register during the TAP controller's UPDATE states. Each of the shift registers is discussed in detail in "Register Descriptions" on page 73.

5.3.4 Operational Considerations

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

5.3.4.1 GPIO Functionality

When the microcontroller is reset with either a POR or RST, the JTAG/SWD port pins default to their JTAG/SWD configurations. The default configuration includes enabling digital functionality (`DEN[3:0]` set in the **Port C GPIO Digital Enable (GPIODEN)** register), enabling the pull-up resistors (`PUE[3:0]` set in the **Port C GPIO Pull-Up Select (GPIOPUR)** register), disabling the pull-down resistors (`PDE[3:0]` cleared in the **Port C GPIO Pull-Down Select (GPIOPDR)** register) and enabling the alternate hardware function (`AFSEL[3:0]` set in the **Port C GPIO Alternate Function Select (GPIOAFSEL)** register) on the JTAG/SWD pins. See page 307, page 313, page 315, and page 318.

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

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

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the `NMI` pin (`PB7`) and the four JTAG/SWD pins (`PC[3:0]`). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 307), **GPIO Pull Up Select (GPIOPUR)** register (see page 313), **GPIO Pull-Down Select (GPIOPDR)** register (see page 315), and **GPIO Digital Enable (GPIODEN)** register (see page 318) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 320) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 321) have been set.

5.3.4.2 Communication with JTAG/SWD

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

5.3.4.3 Recovering a "Locked" Microcontroller

Note: Performing the sequence below restores the nonvolatile registers discussed in “Nonvolatile Register Programming” on page 198 to their factory default values. The mass erase of the Flash memory caused by the sequence below occurs prior to the nonvolatile registers being restored.

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

1. Assert and hold the RST signal.

2. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence on the section called "JTAG-to-SWD Switching" on page 72.
3. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence on the section called "SWD-to-JTAG Switching" on page 73.
4. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
5. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
6. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
7. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
8. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
9. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
10. Perform steps 1 and 2 of the JTAG-to-SWD switch sequence.
11. Perform steps 1 and 2 of the SWD-to-JTAG switch sequence.
12. Release the $\overline{\text{RST}}$ signal.
13. Wait 400 ms.
14. Power-cycle the microcontroller.

5.3.4.4 ARM Serial Wire Debug (SWD)

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

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

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

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

JTAG-to-SWD Switching

To switch the operating mode of the Debug Access Port (DAP) from JTAG to SWD mode, the external debug hardware must send the switching preamble to the microcontroller. The 16-bit TMS command for switching to SWD mode is defined as b1110.0111.1001.1110, transmitted LSB first.

This command can also be represented as `0xE79E` when transmitted LSB first. The complete switch sequence should consist of the following transactions on the `TCK/SWCLK` and `TMS/SWDIO` signals:

1. Send at least 50 `TCK/SWCLK` cycles with `TMS/SWDIO` High to ensure that both JTAG and SWD are in their reset/idle states.
2. Send the 16-bit JTAG-to-SWD switch command, `0xE79E`, on `TMS`.
3. Send at least 50 `TCK/SWCLK` cycles with `TMS/SWDIO` High to ensure that if SWJ-DP was already in SWD mode, the SWD goes into the line reset state before sending the switch sequence.

SWD-to-JTAG Switching

To switch the operating mode of the Debug Access Port (DAP) from SWD to JTAG mode, the external debug hardware must send a switch command to the microcontroller. The 16-bit `TMS` command for switching to JTAG mode is defined as b1110.0111.0011.1100, transmitted LSB first. This command can also be represented as `0xE73C` when transmitted LSB first. The complete switch sequence should consist of the following transactions on the `TCK/SWCLK` and `TMS/SWDIO` signals:

1. Send at least 50 `TCK/SWCLK` cycles with `TMS/SWDIO` High to ensure that both JTAG and SWD are in their reset/idle states.
2. Send the 16-bit SWD-to-JTAG switch command, `0xE73C`, on `TMS`.
3. Send at least 50 `TCK/SWCLK` cycles with `TMS/SWDIO` High to ensure that if SWJ-DP was already in JTAG mode, the JTAG goes into the Test Logic Reset state before sending the switch sequence.

5.4 Initialization and Configuration

After a Power-On-Reset or an external reset ($\overline{\texttt{RST}}$), the JTAG pins are automatically configured for JTAG communication. No user-defined initialization or configuration is needed. However, if the user application changes these pins to their GPIO function, they must be configured back to their JTAG functionality before JTAG communication can be restored. To return the pins to their JTAG functions, enable the four JTAG pins (`PC[3:0]`) for their alternate function using the **GPIOAFSEL** register. In addition to enabling the alternate functions, any other changes to the GPIO pad configurations on the four JTAG pins (`PC[3:0]`) should be returned to their default settings.

5.5 Register Descriptions

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

5.5.1 Instruction Register (IR)

The JTAG TAP Instruction Register (IR) is a four-bit serial scan chain connected between the JTAG `TDI` and `TDO` pins with a parallel load register. When the TAP Controller is placed in the correct states, bits can be shifted into the IR. Once these bits have been shifted into the chain and updated, they are interpreted as the current instruction. The decode of the IR bits is shown in Table 5-3. A detailed explanation of each instruction, along with its associated Data Register, follows.

Table 5-3. JTAG Instruction Register Commands

IR[3:0]	Instruction	Description
0x0	EXTEST	Drives the values preloaded into the Boundary Scan Chain by the SAMPLE/PRELOAD instruction onto the pads.
0x1	INTEST	Drives the values preloaded into the Boundary Scan Chain by the SAMPLE/PRELOAD instruction into the controller.
0x2	SAMPLE / PRELOAD	Captures the current I/O values and shifts the sampled values out of the Boundary Scan Chain while new preload data is shifted in.
0x8	ABORT	Shifts data into the ARM Debug Port Abort Register.
0xA	DPACC	Shifts data into and out of the ARM DP Access Register.
0xB	APACC	Shifts data into and out of the ARM AC Access Register.
0xE	IDCODE	Loads manufacturing information defined by the *IEEE Standard 1149.1* into the IDCODE chain and shifts it out.
0xF	BYPASS	Connects `TDI` to `TDO` through a single Shift Register chain.
All Others	Reserved	Defaults to the BYPASS instruction to ensure that `TDI` is always connected to `TDO`.

5.5.1.1 EXTEST Instruction

The EXTEST instruction is not associated with its own Data Register chain. Instead, the EXTEST instruction uses the data that has been preloaded into the Boundary Scan Data Register using the SAMPLE/PRELOAD instruction. When the EXTEST instruction is present in the Instruction Register, the preloaded data in the Boundary Scan Data Register associated with the outputs and output enables are used to drive the GPIO pads rather than the signals coming from the core. With tests that drive known values out of the controller, this instruction can be used to verify connectivity. While the EXTEST instruction is present in the Instruction Register, the Boundary Scan Data Register can be accessed to sample and shift out the current data and load new data into the Boundary Scan Data Register.

5.5.1.2 INTEST Instruction

The INTEST instruction is not associated with its own Data Register chain. Instead, the INTEST instruction uses the data that has been preloaded into the Boundary Scan Data Register using the SAMPLE/PRELOAD instruction. When the INTEST instruction is present in the Instruction Register, the preloaded data in the Boundary Scan Data Register associated with the inputs are used to drive the signals going into the core rather than the signals coming from the GPIO pads. With tests that drive known values into the controller, this instruction can be used for testing. It is important to note that although the RST input pin is on the Boundary Scan Data Register chain, it is only observable. While the INTEST instruction is present in the Instruction Register, the Boundary Scan Data Register can be accessed to sample and shift out the current data and load new data into the Boundary Scan Data Register.

5.5.1.3 SAMPLE/PRELOAD Instruction

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

While these samples of the inputs, outputs, and output enables are being shifted out of the Boundary Scan Data Register, new data is being shifted into the Boundary Scan Data Register from `TDI`. Once the new data has been shifted into the Boundary Scan Data Register, the data is saved in the parallel load registers when the TAP controller enters the Update DR state. This update of the parallel load register preloads data into the Boundary Scan Data Register that is associated with each input, output, and output enable. This preloaded data can be used with the EXTEST and INTEST instructions to drive data into or out of the controller. See "Boundary Scan Data Register" on page 76 for more information.

5.5.1.4 ABORT Instruction

The ABORT instruction connects the associated ABORT Data Register chain between `TDI` and `TDO`. This instruction provides read and write access to the ABORT Register of the ARM Debug Access Port (DAP). Shifting the proper data into this Data Register clears various error bits or initiates a DAP abort of a previous request. See the "ABORT Data Register" on page 77 for more information.

5.5.1.5 DPACC Instruction

The DPACC instruction connects the associated DPACC Data Register chain between `TDI` and `TDO`. This instruction provides read and write access to the DPACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to the ARM debug and status registers. See "DPACC Data Register" on page 77 for more information.

5.5.1.6 APACC Instruction

The APACC instruction connects the associated APACC Data Register chain between `TDI` and `TDO`. This instruction provides read and write access to the APACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to internal components and buses through the Debug Port. See "APACC Data Register" on page 77 for more information.

5.5.1.7 IDCODE Instruction

The IDCODE instruction connects the associated IDCODE Data Register chain between `TDI` and `TDO`. This instruction provides information on the manufacturer, part number, and version of the ARM core. This information can be used by testing equipment and debuggers to automatically configure input and output data streams. IDCODE is the default instruction loaded into the JTAG Instruction Register when a Power-On-Reset (POR) is asserted, or the Test-Logic-Reset state is entered. See "IDCODE Data Register" on page 76 for more information.

5.5.1.8 BYPASS Instruction

The BYPASS instruction connects the associated BYPASS Data Register chain between `TDI` and `TDO`. This instruction is used to create a minimum length serial path between the `TDI` and `TDO` ports. The BYPASS Data Register is a single-bit shift register. This instruction improves test efficiency by allowing components that are not needed for a specific test to be bypassed in the JTAG scan chain by loading them with the BYPASS instruction. See "BYPASS Data Register" on page 76 for more information.

5.5.2 Data Registers

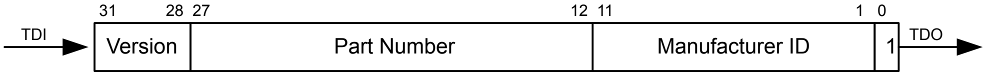
The JTAG module contains six Data Registers. These serial Data Register chains include: IDCODE, BYPASS, Boundary Scan, APACC, DPACC, and ABORT and are discussed in the following sections.

5.5.2.1 IDCODE Data Register

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

The major uses of the JTAG port are for manufacturer testing of component assembly and program development and debug. To facilitate the use of auto-configuration debug tools, the IDCODE instruction outputs a value of 0x4BA0.0477. This value allows the debuggers to automatically configure themselves to work correctly with the Cortex-M3 during debug.

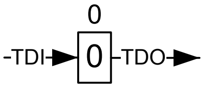
Figure 5-3. IDCODE Register Format



5.5.2.2 BYPASS Data Register

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

Figure 5-4. BYPASS Register Format

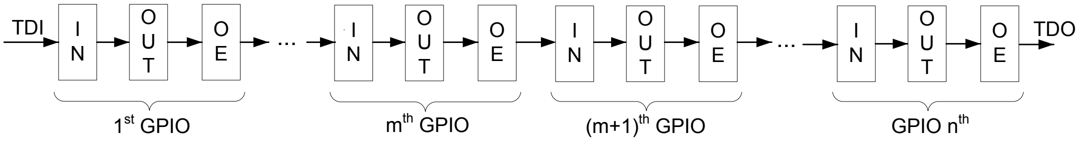


5.5.2.3 Boundary Scan Data Register

The format of the Boundary Scan Data Register is shown in Figure 5-5. Each GPIO pin, starting with a GPIO pin next to the JTAG port pins, is included in the Boundary Scan Data Register. Each GPIO pin has three associated digital signals that are included in the chain. These signals are input, output, and output enable, and are arranged in that order as shown in the figure.

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

Figure 5-5. Boundary Scan Register Format



5.5.2.4 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*.

5.5.2.5 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*.

5.5.2.6 ABORT Data Register

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

6 System Control

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

6.1 Signal Description

Table 6-1 on page 78 lists the external signals of the System Control module and describes the function of each. The `NMI` signal is the alternate function for the GPIO `PB7` signal and functions as a GPIO after reset. `PB7` is under commit protection and requires a special process to be configured as the `NMI` signal or to subsequently return to the GPIO function, see “Commit Control” on page 291. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the `NMI` signal. The `AFSEL` bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 307) should be set to choose the NMI function. The number in parentheses is the encoding that must be programmed into the `PMCn` field in the **GPIO Port Control (GPIOPCTL)** register (page 324) to assign the `NMI` signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 285. The remaining signals (with the word "fixed" in the Pin Mux/Pin Assignment column) have a fixed pin assignment and function.

Table 6-1. Signals for System Control & Clocks

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`NMI`	55	PB7 (4)	I	TTL	Non-maskable interrupt.
`OSC0`	30	fixed	I	Analog	Main oscillator crystal input or an external clock reference input.
`OSC1`	31	fixed	O	Analog	Main oscillator crystal output.
$\overline{\texttt{RST}}$	40	fixed	I	TTL	System reset input.

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

6.2 Functional Description

The System Control module provides the following capabilities:

- Device identification, see “Device Identification” on page 78
- Local control, such as reset (see “Reset Control” on page 78), power (see “Power Control” on page 83) and clock control (see “Clock Control” on page 84)
- System control (Run, Sleep, and Deep-Sleep modes), see “System Control” on page 90

6.2.1 Device Identification

Several read-only registers provide software with information on the microcontroller, such as version, part number, SRAM size, Flash memory size, and other features. See the **DID0** (page 94), **DID1** (page 121), **DC0**-**DC9** (page 123) and **NVMSTAT** (page 140) registers.

6.2.2 Reset Control

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

6.2.2.1 Reset Sources

The LM3S1W16 microcontroller has six sources of reset:

1. Power-on reset (POR) (see page 79).
2. External reset input pin ($\overline{\text{RST}}$) assertion (see page 80).
3. Internal brown-out (BOR) detector (see page 81).
4. Software-initiated reset (with the software reset registers) (see page 81).
5. A watchdog timer reset condition violation (see page 82).
6. MOSC failure (see page 83).

Table 6-2 provides a summary of results of the various reset operations.

Table 6-2. Reset Sources

Reset Source	Core Reset?	JTAG Reset?	On-Chip Peripherals Reset?
Power-On Reset	Yes	Yes	Yes
$\overline{\text{RST}}$	Yes	Pin Config Only	Yes
Brown-Out Reset	Yes	No	Yes
Software Reset	Yes[a]	No	Yes[b]
Watchdog Reset	Yes	No	Yes
MOSC Failure Reset	Yes	No	Yes

a. By using the SYSRESETREQ bit in the ARM Cortex-M3 Application Interrupt and Reset Control register

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

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

6.2.2.2 Power-On Reset (POR)

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

The internal Power-On Reset (POR) circuit monitors the power supply voltage (V_{DD}) and generates a reset signal to all of the internal logic including JTAG when the power supply ramp reaches a threshold value (V_{TH}). The microcontroller must be operating within the specified operating parameters when the on-chip power-on reset pulse is complete. The 3.3-V power supply to the microcontroller must reach 3.0 V within 10 msec of V_{DD} crossing 2.0 V to guarantee proper operation. For applications that require the use of an external reset signal to hold the microcontroller in reset longer than the internal POR, the $\overline{\text{RST}}$ input may be used as discussed in “External $\overline{\text{RST}}$ Pin” on page 80.

The Power-On Reset sequence is as follows:

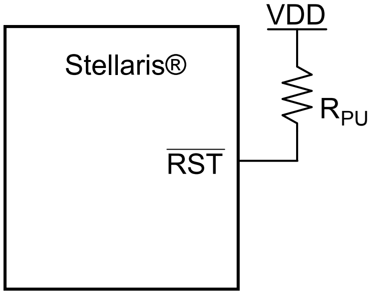
1. The microcontroller waits for internal POR to go inactive.
2. The internal reset is released and the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

The internal POR is only active on the initial power-up of the microcontroller. The Power-On Reset timing is shown in Figure 21-5 on page 651.

6.2.2.3 External $\overline{\text{RST}}$ Pin

If the application only uses the internal POR circuit, the $\overline{\text{RST}}$ input must be connected to the power supply (V_{DD}) through an optional pull-up resistor (0 to 10K Ω) as shown in Figure 6-1 on page 80.

Figure 6-1. Basic $\overline{\text{RST}}$ Configuration



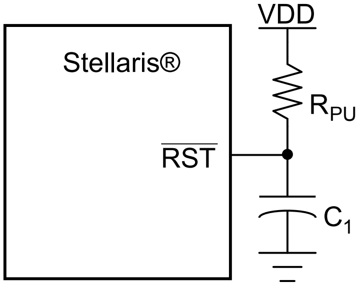
R_{PU} = 0 to 100 kΩ

The external reset pin ($\overline{\text{RST}}$) resets the microcontroller including the core and all the on-chip peripherals except the JTAG TAP controller (see “JTAG Interface” on page 66). The external reset sequence is as follows:

1. The external reset pin ($\overline{\text{RST}}$) is asserted for the duration specified by T_{MIN} and then de-asserted (see “Reset” on page 650).
2. The internal reset is released and the core loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

To improve noise immunity and/or to delay reset at power up, the $\overline{\text{RST}}$ input may be connected to an RC network as shown in Figure 6-2 on page 80.

Figure 6-2. External Circuitry to Extend Power-On Reset

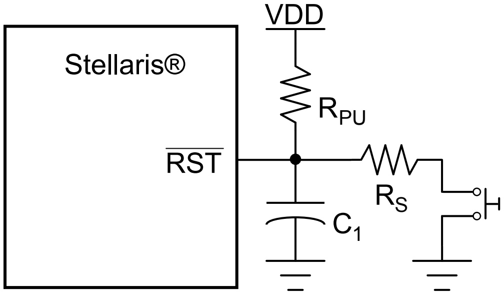


R_{PU} = 1 kΩ to 100 kΩ

C_1 = 1 nF to 10 µF

If the application requires the use of an external reset switch, Figure 6-3 on page 81 shows the proper circuitry to use.

Figure 6-3. Reset Circuit Controlled by Switch



Typical R_{PU} = 10 kΩ

Typical R_S = 470 Ω

C_1 = 10 nF

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

The external reset timing is shown in Figure 21-4 on page 650.

6.2.2.4 Brown-Out Reset (BOR)

The microcontroller provides a brown-out detection circuit that triggers if the power supply (V_{DD}) drops below a brown-out threshold voltage (V_{BTH}). If a brown-out condition is detected, the system may generate an interrupt or a system reset. Brown-out resets are controlled with the **Power-On and Brown-Out Reset Control (PBORCTL)** register. The `BORIOR` bit in the **PBORCTL** register must be set for a brown-out condition to trigger a reset; if `BORIOR` is clear, an interrupt is generated. The default condition is to generate an interrupt, so BOR must be enabled. When a Brown-out condition occurs during a Flash PROGRAM or ERASE operation, a full system reset is always triggered without regard to the setting in the **PBORCTL** register.

The result of a brown-out reset is equivalent to that of an assertion of the external $\overline{\text{RST}}$ 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 21-6 on page 651.

6.2.2.5 Software Reset

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

Peripherals can be individually reset by software via three registers that control reset signals to each on-chip peripheral (see the **SRCRn** registers, page 162). If the bit position corresponding to a peripheral is set and subsequently cleared, the peripheral is reset. The encoding of the reset registers is consistent with the encoding of the clock gating control for peripherals and on-chip functions (see “System Control” on page 90).

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

1. A software microcontroller reset is initiated by setting the SYSRESETREQ bit in the ARM Cortex-M3 Application Interrupt and Reset Control register.

2. An internal reset is asserted.
3. The internal reset is deasserted and the microcontroller loads from memory the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

The software-initiated system reset timing is shown in Figure 21-7 on page 651.

6.2.2.6 Watchdog Timer Reset

The Watchdog Timer module's function is to prevent system hangs. The LM3S1W16 microcontroller has two Watchdog Timer modules in case one watchdog clock source fails. One watchdog is run off the system clock and the other is run off the Precision Internal Oscillator (PIOSC). Each module operates in the same manner except that because the PIOSC watchdog timer module is in a different clock domain, register accesses must have a time delay between them. The watchdog timer can be configured to generate an interrupt to the microcontroller on its first time-out and to generate a reset on its second time-out.

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

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

For more information on the Watchdog Timer module, see “Watchdog Timers” on page 381.

The watchdog reset timing is shown in Figure 21-8 on page 651.

6.2.3 Non-Maskable Interrupt

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

- The assertion of the `NMI` signal
- A main oscillator verification error
- The `NMISET` bit in the **Interrupt Control and Status (ICSR)** register in the Cortex-M3.

Software must check the cause of the interrupt in order to distinguish among the sources.

6.2.3.1 NMI Pin

The alternate function to GPIO port pin B7 is an NMI signal. The alternate function must be enabled in the GPIO for the signal to be used as an interrupt, as described in “General-Purpose Input/Outputs (GPIOs)” on page 285. Note that enabling the NMI alternate function requires the use of the GPIO lock and commit function just like the GPIO port pins associated with JTAG/SWD functionality, see page 321. The active sense of the NMI signal is High; asserting the enabled NMI signal above V_{IH} initiates the NMI interrupt sequence.

6.2.3.2 Main Oscillator Verification Failure

The LM3S1W16 microcontroller provides a main oscillator verification circuit that generates an error condition if the oscillator is running too fast or two slow. The main oscillator verification circuit can be programmed to generate a reset event, at which time a Power-on Reset is generated and control is transferred to the NMI handler. The NMI handler is used to address the main oscillator verification failure because the necessary code can be removed from the general reset handler, speeding up reset processing. The detection circuit is enabled by setting the `CVAL` bit in the **Main Oscillator Control (MOSCCTL)** register. The main oscillator verification error is indicated in the main oscillator fail status (`MOSCFAIL`) bit in the **Reset Cause (RESC)** register. The main oscillator verification circuit action is described in more detail in “Main Oscillator Verification Circuit” on page 90.

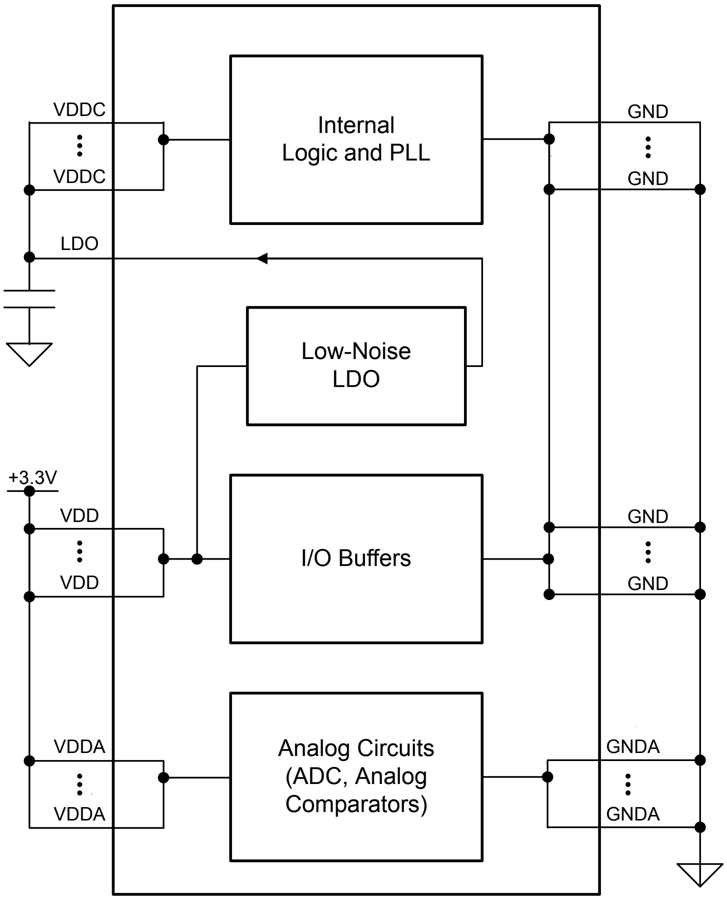
6.2.4 Power Control

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

Figure 6-4 shows the power architecture.

Note: On the printed circuit board, use the `LDO` output as the source of `VDDC` input. In addition, the LDO requires decoupling capacitors. See “On-Chip Low Drop-Out (LDO) Regulator Characteristics” on page 644.

Figure 6-4. Power Architecture



6.2.5 Clock Control

System control determines the control of clocks in this part.

6.2.5.1 Fundamental Clock Sources

There are multiple clock sources for use in the microcontroller:

- **Precision Internal Oscillator (PIOSC).** The precision internal oscillator is an on-chip clock source that is the clock source the microcontroller uses during and following POR. It does not require the use of any external components and provides a clock that is 16 MHz ±1% at room temperature and ±3% across temperature. The PIOSC allows for a reduced system cost in applications that require an accurate clock source. If the main oscillator is required, software must enable the main oscillator following reset and allow the main oscillator to stabilize before changing the clock reference. If the Hibernation Module clock source is a 32.768-kHz oscillator, the precision internal oscillator can be trimmed by software based on a reference clock for increased accuracy.

- **Main Oscillator (MOSC).** The main oscillator provides a frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the `OSC0` input pin, or an external crystal is connected across the `OSC0` input and `OSC1` output pins. If the PLL is being used, the crystal value must be one of the supported frequencies between 3.579545 MHz through

16.384 MHz (inclusive). If the PLL is not being used, the crystal may be any one of the supported frequencies between 1 MHz and 16.384 MHz. The single-ended clock source range is from DC through the specified speed of the microcontroller. The supported crystals are listed in the `XTAL` bit field in the **RCC** register (see page 105).

- **Internal 30-kHz Oscillator.** The internal 30-kHz oscillator provides an operational frequency of 30 kHz ± 50%. It is intended for use during Deep-Sleep power-saving modes. This power-savings mode benefits from reduced internal switching and also allows the MOSC and PIOSC to be powered down.
- **Hibernation Module Clock Source.** The Hibernation module can be clocked in one of two ways. The first way is a 4.194304-MHz crystal connected to the `XOSC0` and `XOSC1` pins. This clock signal is divided by 128 internally to produce the 32.768-kHz clock reference. The second way is a 32.768-kHz oscillator connected to the `XOSC0` pin. The clock source for the Hibernation module can be used for the system clock, thus eliminating the need for an additional crystal or oscillator. In addition, a 4.194304-MHz crystal can also be a source for the PLL. The Hibernation module clock source is intended to provide the system with a real-time clock source and may also provide an accurate source of Deep-Sleep or Hibernate mode power savings.

The internal system clock (SysClk), is derived from any of the above sources plus two others: the output of the main internal PLL and the precision internal oscillator divided by four (4 MHz ± 1%). The frequency of the PLL clock reference must be in the range of 3.579545 MHz to 16.384 MHz (inclusive). Table 6-3 on page 85 shows how the various clock sources can be used in a system.

Table 6-3. Clock Source Options

Clock Source	Drive PLL?		Used as SysClk?	
Precision Internal Oscillator	Yes	BYPASS = 0, OSCSRC = 0x1	Yes	BYPASS = 1, OSCSRC = 0x1
Precision Internal Oscillator divide by 4 (4 MHz ± 1%)	No	BYPASS = 1	Yes	BYPASS = 1, OSCSRC = 0x2
Main Oscillator	Yes	BYPASS = 0, OSCSRC = 0x0	Yes	BYPASS = 1, OSCSRC = 0x0
Internal 30-kHz Oscillator	No	BYPASS = 1	Yes	BYPASS = 1, OSCSRC = 0x3
Hibernation Module 4.194304-MHz Crystal	Yes	BYPASS = 0, OSCSRC2 = 0x7	Yes	BYPASS = 1, OSCSRC2 = 0x6
Hibernation Module 32.768-kHz Oscillator	No	BYPASS = 1	Yes	BYPASS = 1, OSCSRC2 = 0x7

6.2.5.2 Clock Configuration

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

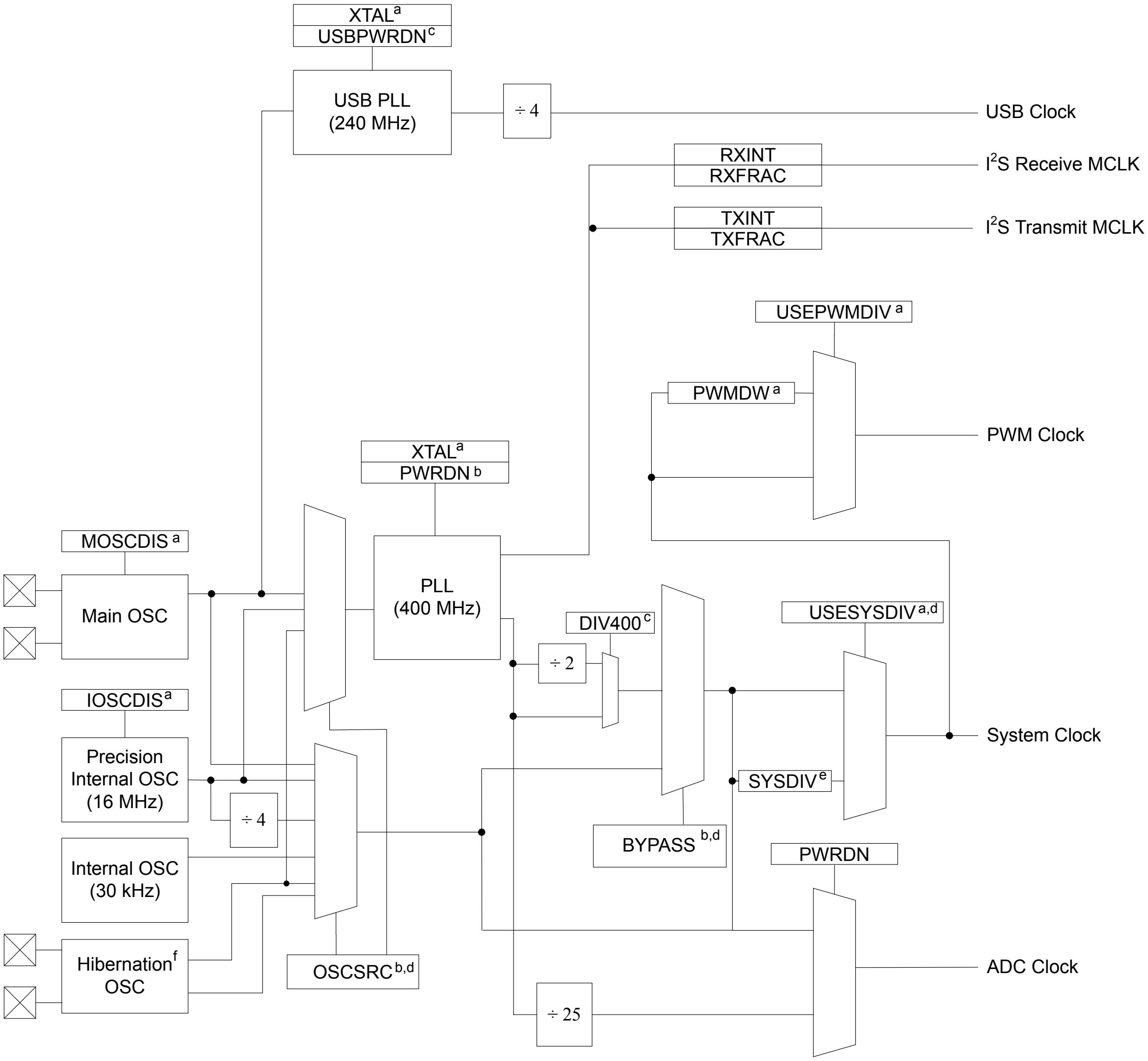
- Source of clocks in sleep and deep-sleep modes
- System clock derived from PLL or other clock source
- Enabling/disabling of oscillators and PLL

- Clock divisors
- Crystal input selection

Figure 6-5 shows the logic for the main clock tree. The peripheral blocks are driven by the system clock signal and can be individually enabled/disabled. The ADC clock signal is automatically divided down to 16 MHz for proper ADC operation.

Note: When the ADC module is in operation, the system clock must be at least 16 MHz.

Figure 6-5. Main Clock Tree



a. Control provided by **RCC** register bit/field.
b. Control provided by **RCC** register bit/field or **RCC2** register bit/field, if overridden with **RCC2** register bit `USERCC2`.
c. Control provided by **RCC2** register bit/field.
d. Also may be controlled by **DSLPCLKCFG** when in deep sleep mode.
e. Control provided by **RCC** register `SYSDIV` field, **RCC2** register `SYSDIV2` field if overridden with `USERCC2` bit, or [`SYSDIV2`,`SYSDIV2LSB`] if both `USERCC2` and `DIV400` bits are set.
f. Only a 4.194304-Mhz crystal can be used to drive the PLL.

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

In the **RCC** register, the `SYSDIV` field specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the `BYPASS` bit in this register is configured). When using the PLL, the VCO frequency of 400 MHz is predivided by 2 before the divisor is applied. Table 6-4 shows how the `SYSDIV` encoding affects the system clock frequency, depending on whether the PLL is used (`BYPASS=0`) or another clock source is used (`BYPASS=1`). The divisor is equivalent to the `SYSDIV` encoding plus 1. For a list of possible clock sources, see Table 6-3 on page 85.

Table 6-4. Possible System Clock Frequencies Using the SYSDIV Field

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

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

The `SYSDIV2` field in the **RCC2** register is 2 bits wider than the `SYSDIV` field in the **RCC** register so that additional larger divisors up to /64 are possible, allowing a lower system clock frequency for improved Deep Sleep power consumption. When using the PLL, the VCO frequency of 400 MHz is predivided by 2 before the divisor is applied. The divisor is equivalent to the `SYSDIV2` encoding plus 1. Table 6-5 shows how the `SYSDIV2` encoding affects the system clock frequency, depending on whether the PLL is used (`BYPASS2=0`) or another clock source is used (`BYPASS2=1`). For a list of possible clock sources, see Table 6-3 on page 85.

Table 6-5. Examples of Possible System Clock Frequencies Using the SYSDIV2 Field

SYSDIV2	Divisor	Frequency (BYPASS2=0)	Frequency (BYPASS2=1)	StellarisWare Parameter[a]
0x00	/1	reserved	Clock source frequency	SYSCTL_SYSDIV_1
0x01	/2	reserved	Clock source frequency/2	SYSCTL_SYSDIV_2
0x02	/3	reserved	Clock source frequency/3	SYSCTL_SYSDIV_3
0x03	/4	50 MHz	Clock source frequency/4	SYSCTL_SYSDIV_4
...	...	...	...	...
0x09	/10	20 MHz	Clock source frequency/10	SYSCTL_SYSDIV_10
...	...	...	...	...

Table 6-5. Examples of Possible System Clock Frequencies Using the SYSDIV2 Field *(continued)*

SYSDIV2	Divisor	Frequency (BYPASS2=0)	Frequency (BYPASS2=1)	StellarisWare Parameter[a]
0x3F	/64	3.125 MHz	Clock source frequency/64	SYSCTL_SYSDIV_64

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

To allow for additional frequency choices when using the PLL, the `DIV400` bit is provided along with the `SYSDIV2LSB` bit. When the `DIV400` bit is set, bit 22 becomes the LSB for `SYSDIV2`. In this situation, the divisor is equivalent to the (`SYSDIV2` encoding with `SYSDIV2LSB` appended) plus one. When the `DIV400` bit is clear, `SYSDIV2LSB` is ignored, and the system clock frequency is determined as shown in Table 6-5 on page 87. Care must be taken when using these frequency choices with StellarisWare DriverLib API functions. see Table 6-6.

Table 6-6. Examples of Possible System Clock Frequencies with DIV400=1

SYSDIV2	SYSDIV2LSB	Divisor	Frequency (BYPASS2=0)[a]	StellarisWare Parameter[b]
0x00	reserved	/2	reserved	-
0x01	0	/3	reserved	-
	1	/4	reserved	-
0x02	0	/5	reserved	-
	1	/6	reserved	-
0x03	0	/7	reserved	-
	1	/8	50 MHz	SYSCTL_SYSDIV_4
0x04	0	/9	44.44 MHz	SYSCTL_SYSDIV_4_5
	1	/10	40 MHz	SYSCTL_SYSDIV_5
...	...	...	...	...
0x3F	0	/127	3.15 MHz	SYSCTL_SYSDIV_63_5
	1	/128	3.125 MHz	SYSCTL_SYSDIV_64

a. Note that `DIV400` and `SYSDIV2LSB` are only valid when `BYPASS2`=0.

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

6.2.5.3 Precision Internal Oscillator Operation (PIOSC)

The microcontroller powers up with the PIOSC running. If another clock source is desired, the PIOSC can be powered down by setting the `IOSCDIS` bit in the **RCC** register.

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

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

- Automatic calibration using the enable 32.768-kHz oscillator from the Hibernation module: set the `CAL` bit; the results of the calibration are shown in the `RESULT` field in the **Precision Internal Oscillator Statistic (PIOSCSTAT)** register. After calibration is complete, the PIOSC is trimmed using trimmed value returned in the `CT` field.

6.2.5.4 Crystal Configuration for the Main Oscillator (MOSC)

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

The `XTAL` bit in the **RCC** register (see page 105) 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.

6.2.5.5 Main PLL Frequency Configuration

The main PLL is disabled by default during power-on reset and is enabled later by software if required. Software specifies the output divisor to set the system clock frequency and enables the main PLL to drive the output. The PLL operates at 400 MHz, but is divided by two prior to the application of the output divisor. Table 21-9 on page 646 shows the actual PLL frequency and error for a given crystal choice.

To configure the PIOSC to be the clock source for the main PLL, program the `OSCRC2` field in the **Run-Mode Clock Configuration 2 (RCC2)** register to be 0x1.

If the main oscillator provides the clock reference to the main PLL, the translation provided by hardware and used to program the PLL is available for software in the **XTAL to PLL Translation (PLLCFG)** register (see page 109). The internal translation provides a translation within ± 1% of the targeted PLL VCO frequency.

To configure the Hibernation module clock source as the PLL input reference, program the `OSCRC2` field in the **Run-Mode Clock Configuration 2 (RCC2)** register to be 0x6 for a 4.194304-MHz crystal or 0x7 for an external 32.768-kHz oscillator.

The Crystal Value field (`XTAL`) in the **Run-Mode Clock Configuration (RCC)** register (see page 105) describes the available crystal choices and default programming of the **PLLCFG** register. Any time the `XTAL` field changes, the new settings are translated and the internal PLL settings are updated.

6.2.5.6 PLL Modes

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 105 and page 112).

6.2.5.7 PLL Operation

If a PLL configuration is changed, the PLL output frequency is unstable until it reconverges (relocks) to the new setting. The time between the configuration change and relock is T_{READY} (see Table 21-8 on page 646). During the relock time, the affected PLL is not usable as a clock reference.

PLL is changed by one of the following:

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

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

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

6.2.5.8 Main Oscillator Verification Circuit

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

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

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

6.2.6 System Control

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

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

- **Run Mode.** In Run mode, the microcontroller actively executes code. Run mode provides normal operation of the processor and all of the peripherals that are currently enabled by the **RCGCn** registers. The system clock can be any of the available clock sources including the PLL.
- **Sleep Mode.** In Sleep mode, the clock frequency of the active peripherals is unchanged, but the processor and the memory subsystem are not clocked and therefore no longer execute code. Sleep mode is entered by the Cortex-M3 core executing a WFI (Wait for Interrupt) instruction.

Any properly configured interrupt event in the system brings the processor back into Run mode. See the system control NVIC section of the *ARM® Cortex™-M3 Technical Reference Manual* for more details.

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

- **Deep-Sleep Mode.** In Deep-Sleep mode, the clock frequency of the active peripherals may change (depending on the Run mode clock configuration) in addition to the processor clock being stopped. An interrupt returns the microcontroller to Run mode from one of the sleep modes; the sleep modes are entered on request from the code. Deep-Sleep mode is entered by first writing the Deep Sleep Enable bit in the ARM Cortex-M3 NVIC system control register and then executing a WFI instruction. Any properly configured interrupt event in the system brings the processor back into Run mode. See the system control NVIC section of the *ARM® Cortex™-M3 Technical Reference Manual* for more details.

  The Cortex-M3 processor core and the memory subsystem are not clocked. Peripherals are clocked that are enabled in the **DCGCn** register when auto-clock gating is enabled (see the **RCC** register) or the **RCGCn** register when auto-clock gating is disabled. The system clock source is specified in the **DSLPCLKCFG** register. When the **DSLPCLKCFG** register is used, the internal oscillator source is powered up, if necessary, and other clocks are powered down. If the PLL is running at the time of the WFI instruction, hardware powers the PLL down and overrides the `SYSDIV` field of the active **RCC**/**RCC2** register, to be determined by the `DSDIVORIDE` setting in the **DSLPCLKCFG** register, up to /16 or /64 respectively. When the Deep-Sleep exit event occurs, hardware brings the system clock back to the source and frequency it had at the onset of Deep-Sleep mode before enabling the clocks that had been stopped during the Deep-Sleep duration. If the PIOSC or the 4.194304-MHz Hibernation module clock source is used as the PLL reference clock source, it may continue to provide the clock during Deep-Sleep. See page 116.

- **Hibernate Mode.** In this mode, the power supplies are turned off to the main part of the microcontroller and only the Hibernation module's circuitry is active. An external wake event or RTC event is required to bring the microcontroller back to Run mode. The Cortex-M3 processor and peripherals outside of the Hibernation module see a normal "power on" sequence and the processor starts running code. Software can determine if the microcontroller has been restarted from Hibernate mode by inspecting the Hibernation module registers.

6.3 Initialization and Configuration

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

1. Bypass the PLL and system clock divider by setting the `BYPASS` bit and clearing the `USESYS` bit in the **RCC** register, thereby configuring the microcontroller to run off a “raw” clock source and allowing for the new PLL configuration to be validated before switching the system clock to the PLL.

2. Select the crystal value (`XTAL`) and oscillator source (`OSCSRC`), and clear the `PWRDN` bit in **RCC**/**RCC2**. Setting the `XTAL` field automatically pulls valid PLL configuration data for the appropriate crystal, and clearing the `PWRDN` bit powers and enables the PLL and its output.

3. Select the desired system divider (`SYSDIV`) in **RCC**/**RCC2** and set the `USESYS` bit in **RCC**. The `SYSDIV` field determines the system frequency for the microcontroller.

4. Wait for the PLL to lock by polling the `PLLLRIS` bit in the **Raw Interrupt Status (RIS**) register.

5. Enable use of the PLL by clearing the `BYPASS` bit in **RCC**/**RCC2**.

6.4 Register Map

Table 6-7 on page 92 lists the System Control registers, grouped by function. The offset listed is a hexadecimal increment to the register's address, relative to the System Control base address of 0x400F.E000.

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

Additional Flash and ROM registers defined in the System Control register space are described in the "Internal Memory" on page 194.

Table 6-7. System Control Register Map

Offset	Name	Type	Reset	Description	See page
0x000	DID0	RO	-	Device Identification 0	94
0x004	DID1	RO	-	Device Identification 1	121
0x008	DC0	RO	0x001F.000F	Device Capabilities 0	123
0x010	DC1	RO	-	Device Capabilities 1	124
0x014	DC2	RO	0x0307.5037	Device Capabilities 2	126
0x018	DC3	RO	0xBFFF.0FC0	Device Capabilities 3	128
0x01C	DC4	RO	0x0004.301F	Device Capabilities 4	130
0x020	DC5	RO	0x0000.0000	Device Capabilities 5	132
0x024	DC6	RO	0x0000.0000	Device Capabilities 6	133
0x028	DC7	RO	0xFFFF.FFFF	Device Capabilities 7	134
0x02C	DC8	RO	0x0000.00FF	Device Capabilities 8 ADC Channels	138
0x030	PBORCTL	R/W	0x0000.7FFD	Brown-Out Reset Control	96
0x040	SRCR0	R/W	0x00000000	Software Reset Control 0	162
0x044	SRCR1	R/W	0x00000000	Software Reset Control 1	164
0x048	SRCR2	R/W	0x00000000	Software Reset Control 2	166
0x050	RIS	RO	0x0000.0000	Raw Interrupt Status	97
0x054	IMC	R/W	0x0000.0000	Interrupt Mask Control	99
0x058	MISC	R/W1C	0x0000.0000	Masked Interrupt Status and Clear	101
0x05C	RESC	R/W	-	Reset Cause	103
0x060	RCC	R/W	0x0780.3AD1	Run-Mode Clock Configuration	105
0x064	PLLCFG	RO	-	XTAL to PLL Translation	109
0x06C	GPIOHBCTL	R/W	0x0000.0000	GPIO High-Performance Bus Control	110

Table 6-7. System Control Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x070	RCC2	R/W	0x0780.6810	Run-Mode Clock Configuration 2	112
0x07C	MOSCCTL	R/W	0x0000.0000	Main Oscillator Control	115
0x100	RCGC0	R/W	0x00000040	Run Mode Clock Gating Control Register 0	141
0x104	RCGC1	R/W	0x00000000	Run Mode Clock Gating Control Register 1	147
0x108	RCGC2	R/W	0x00000000	Run Mode Clock Gating Control Register 2	156
0x110	SCGC0	R/W	0x00000040	Sleep Mode Clock Gating Control Register 0	143
0x114	SCGC1	R/W	0x00000000	Sleep Mode Clock Gating Control Register 1	150
0x118	SCGC2	R/W	0x00000000	Sleep Mode Clock Gating Control Register 2	158
0x120	DCGC0	R/W	0x00000040	Deep Sleep Mode Clock Gating Control Register 0	145
0x124	DCGC1	R/W	0x00000000	Deep-Sleep Mode Clock Gating Control Register 1	153
0x128	DCGC2	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 2	160
0x144	DSLPCLKCFG	R/W	0x0780.0000	Deep Sleep Clock Configuration	116
0x150	PIOSCCAL	R/W	0x0000.0000	Precision Internal Oscillator Calibration	118
0x154	PIOSCSTAT	RO	0x0000.0040	Precision Internal Oscillator Statistics	120
0x190	DC9	RO	0x0000.00FF	Device Capabilities 9 ADC Digital Comparators	139
0x1A0	NVMSTAT	RO	0x0000.0001	Non-Volatile Memory Information	140

6.5 Register Descriptions

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

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

This register identifies the version of the microcontroller.

Device Identification 0 (DID0)

Base 0x400F.E000
Offset 0x000
Type RO, reset -

	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	1	0	0

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

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

Register 2: 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	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
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORIOR	R/W	0	BOR Interrupt or Reset Value Description 0 A Brown Out Event causes an interrupt to be generated to the interrupt controller. 1 A Brown Out Event causes a reset of the microcontroller.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 3: Raw Interrupt Status (RIS), offset 0x050

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

Raw Interrupt Status (RIS)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved							MOSCPUPRIS	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
31:9	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	MOSCPUPRIS	RO	0	MOSC Power Up Raw Interrupt Status Value Description 1 Sufficient time has passed for the MOSC to reach the expected frequency. The value for this power-up time is indicated by T_{MOSC_SETTLE}. 0 Sufficient time has not passed for the MOSC to reach the expected frequency. This bit is cleared by writing a 1 to the `MOSCPUPMIS` bit in the **MISC** register.
7	reserved	RO	0	Software should not rely on the value of 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 Value Description 1 The PLL timer has reached T_{READY} indicating that sufficient time has passed for the PLL to lock. 0 The PLL timer has not reached T_{READY}. This bit is cleared by writing a 1 to the `PLLLMIS` bit in the **MISC** register.
5:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
1	BORRIS	RO	0	Brown-Out Reset Raw Interrupt Status Value Description 1 A brown-out condition is currently active. 0 A brown-out condition is not currently active. Note the `BORIOR` bit in the **PBORCTL** register must be cleared to cause an interrupt due to a Brown Out Event. This bit is cleared by writing a 1 to the `BORMIS` bit in the **MISC** register.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

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

Interrupt Mask Control (IMC)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved							MOSCPUPIM	reserved	PLLLIM	reserved				BORIM	reserved
Type	RO	RO	RO	RO	RO	RO	RO	R/W	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

Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	MOSCPUPIM	R/W	0	MOSC Power Up Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `MOSCPUPRIS` bit in the **RIS** register is set. 0 The `MOSCPUPRIS` interrupt is suppressed and not sent to the interrupt controller.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PLLLIM	R/W	0	PLL Lock Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `PLLLRIS` bit in the **RIS** register is set. 0 The `PLLLRIS` interrupt is suppressed and not sent to the interrupt controller.
5:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
1	BORIM	R/W	0	Brown-Out Reset Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `BORRIS` bit in the **RIS** register is set. 0 The `BORRIS` interrupt is suppressed and not sent to the interrupt controller.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 5: Masked Interrupt Status and Clear (MISC), offset 0x058

On a read, this register gives the current masked status value of the corresponding interrupt in the **Raw Interrupt Status (RIS)** register. All of the bits are R/W1C, thus writing a 1 to a bit clears the corresponding raw interrupt bit in the **RIS** register (see page 97).

Masked Interrupt Status and Clear (MISC)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved							MOSCPUPMIS	reserved	PLLLMIS	reserved				BORMIS	reserved
Type	RO	RO	RO	RO	RO	RO	RO	R/W1C	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
31:9	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	MOSCPUPMIS	R/W1C	0	MOSC Power Up Masked Interrupt Status Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the MOSC PLL to lock. Writing a 1 to this bit clears it and also the `MOSCPUPRIS` bit in the **RIS** register. 0 When read, a 0 indicates that sufficient time has not passed for the MOSC PLL to lock. A write of 0 has no effect on the state of this bit.
7	reserved	RO	0	Software should not rely on the value of 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 Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because sufficient time has passed for the PLL to lock. Writing a 1 to this bit clears it and also the `PLLLRIS` bit in the **RIS** register. 0 When read, a 0 indicates that sufficient time has not passed for the PLL to lock. A write of 0 has no effect on the state of this bit.

Bit/Field	Name	Type	Reset	Description
5:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORMIS	R/W1C	0	BOR Masked Interrupt Status Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because of a brown-out condition. Writing a 1 to this bit clears it and also the `BORRIS` bit in the **RIS** register. 0 When read, a 0 indicates that a brown-out condition has not occurred. A write of 0 has no effect on the state of this bit.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 6: Reset Cause (RESC), offset 0x05C

This register is set with the reset cause after reset. The bits in this register are sticky and maintain their state across multiple reset sequences, except when an power-on reset is the cause, in which case, all bits other than `POR` in the **RESC** register are cleared.

Reset Cause (RESC)

Base 0x400F.E000
Offset 0x05C
Type R/W, reset -

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										WDT1	SW	WDT0	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
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	MOSCFAIL	R/W	-	MOSC Failure Reset Value Description 1 When read, this bit indicates that the MOSC circuit was enabled for clock validation and failed, generating a reset event. 0 When read, this bit indicates that a MOSC failure has not generated a reset since the previous power-on reset. Writing a 0 to this bit clears it.
15:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	WDT1	R/W	-	Watchdog Timer 1 Reset Value Description 1 When read, this bit indicates that Watchdog Timer 1 timed out and generated a reset. 0 When read, this bit indicates that Watchdog Timer 1 has not generated a reset since the previous power-on reset. Writing a 0 to this bit clears it.

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

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

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

Run-Mode Clock Configuration (RCC)

Base 0x400F.E000
Offset 0x060
Type R/W, reset 0x0780.3AD1

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

Bit/Field	Name	Type	Reset	Description
31:28	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
27	ACG	R/W	0	Auto Clock Gating This bit specifies whether the system uses the **Sleep-Mode Clock Gating Control (SCGCn)** registers and **Deep-Sleep-Mode Clock Gating Control (DCGCn)** registers if the microcontroller enters a Sleep or Deep-Sleep mode (respectively). Value Description 1 The **SCGCn** or **DCGCn** registers are used to control the clocks distributed to the peripherals when the microcontroller is in a sleep mode. The **SCGCn** and **DCGCn** registers allow unused peripherals to consume less power when the microcontroller is in a sleep mode. 0 The **Run-Mode Clock Gating Control (RCGCn)** registers are used when the microcontroller enters a sleep mode. The **RCGCn** registers are always used to control the clocks in Run mode.
26:23	SYSDIV	R/W	0xF	System Clock Divisor Specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the `BYPASS` bit in this register is configured). See Table 6-4 on page 87 for bit encodings. If the `SYSDIV` value is less than `MINSYSDIV` (see page 124), and the PLL is being used, then the `MINSYSDIV` value is used as the divisor. If the PLL is not being used, the `SYSDIV` value can be less than `MINSYSDIV`.

Bit/Field	Name	Type	Reset	Description
22	USESYSDIV	R/W	0	Enable System Clock Divider Value Description 1 The system clock divider is the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source. If the `USERCC2` bit in the **RCC2** register is set, then the `SYSDIV2` field in the **RCC2** register is used as the system clock divider rather than the `SYSDIV` field in this register. 0 The system clock is used undivided.
21:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	PWRDN	R/W	1	PLL Power Down Value Description 1 The PLL is powered down. Care must be taken to ensure that another clock source is functioning and that the `BYPASS` bit is set before setting this bit. 0 The PLL is operating normally.
12	reserved	RO	1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	BYPASS	R/W	1	PLL Bypass Value Description 1 The system clock is derived from the OSC source and divided by the divisor specified by `SYSDIV`. 0 The system clock is the PLL output clock divided by the divisor specified by `SYSDIV`. See Table 6-4 on page 87 for programming guidelines. **Note:** The ADC must be clocked from the PLL or directly from a 16-MHz clock source to operate properly.

Bit/Field	Name	Type	Reset	Description
10:6	XTAL	R/W	0x0B	Crystal Value This field specifies the crystal value attached to the main oscillator. The encoding for this field is provided below. Depending on the crystal used, the PLL frequency may not be exactly 400 MHz, see Table 21-9 on page 646 for more information.

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

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

Register 8: XTAL to PLL Translation (PLLCFG), offset 0x064

This register provides a means of translating external crystal frequencies into the appropriate PLL settings. This register is initialized during the reset sequence and updated anytime that the XTAL field changes in the **Run-Mode Clock Configuration (RCC)** register (see page 105).

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

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

XTAL to PLL Translation (PLLCFG)

Base 0x400F.E000
Offset 0x064
Type RO, reset -

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		F									R				
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13:5	F	RO	-	PLL F Value This field specifies the value supplied to the PLL's F input.
4:0	R	RO	-	PLL R Value This field specifies the value supplied to the PLL's R input.

Register 9: GPIO High-Performance Bus Control (GPIOHBCTL), offset 0x06C

This register controls which internal bus is used to access each GPIO port. When a bit is clear, the corresponding GPIO port is accessed across the legacy Advanced Peripheral Bus (APB) bus and through the APB memory aperture. When a bit is set, the corresponding port is accessed across the Advanced High-Performance Bus (AHB) bus and through the AHB memory aperture. Each GPIO port can be individually configured to use AHB or APB, but may be accessed only through one aperture. The AHB bus provides better back-to-back access performance than the APB bus. The address aperture in the memory map changes for the ports that are enabled for AHB access (see Table 10-6 on page 294).

GPIO High-Performance Bus Control (GPIOHBCTL)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											PORTE	PORTD	PORTC	PORTB	PORTA
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	PORTE	R/W	0	Port E Advanced High-Performance Bus This bit defines the memory aperture for Port E. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
3	PORTD	R/W	0	Port D Advanced High-Performance Bus This bit defines the memory aperture for Port D. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
2	PORTC	R/W	0	Port C Advanced High-Performance Bus This bit defines the memory aperture for Port C. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.

Bit/Field	Name	Type	Reset	Description
1	PORTB	R/W	0	Port B Advanced High-Performance Bus This bit defines the memory aperture for Port B. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.
0	PORTA	R/W	0	Port A Advanced High-Performance Bus This bit defines the memory aperture for Port A. Value Description 1 Advanced High-Performance Bus (AHB) 0 Advanced Peripheral Bus (APB). This bus is the legacy bus.

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

This register overrides the **RCC** equivalent register fields, as shown in Table 6-8, when the `USERCC2` bit is set, allowing the extended capabilities of the **RCC2** register to be used while also providing a means to be backward-compatible to previous parts. Each **RCC2** field that supersedes an **RCC** field is located at the same LSB bit position; however, some **RCC2** fields are larger than the corresponding **RCC** field.

Table 6-8. RCC2 Fields that Override RCC fields

RCC2 Field...	Overrides RCC Field
`SYSDIV2`, bits[28:23]	`SYSDIV`, bits[26:23]
`PWRDN2`, bit[13]	`PWRDN`, bit[13]
`BYPASS2`, bit[11]	`BYPASS`, bit[11]
`OSCSRC2`, bits[6:4]	`OSCSRC`, bits[5:4]

Run-Mode Clock Configuration 2 (RCC2)

Base 0x400F.E000
Offset 0x070
Type R/W, reset 0x0780.6810

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	USERCC2	DIV400	reserved	SYSDIV2						SYSDIV2LSB	reserved					
Type	R/W	R/W	RO	R/W	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		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	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	USERCC2	R/W	0	Use **RCC2** Value Description 1 The **RCC2** register fields override the **RCC** register fields. 0 The **RCC** register fields are used, and the fields in **RCC2** are ignored.
30	DIV400	R/W	0	Divide PLL as 400 MHz vs. 200 MHz This bit, along with the `SYSDIV2LSB` bit, allows additional frequency choices. Value Description 1 Append the `SYSDIV2LSB` bit to the `SYSDIV2` field to create a 7 bit divisor using the 400 MHz PLL output, see Table 6-6 on page 88. 0 Use `SYSDIV2` as is and apply to 200 MHz predivided PLL output. See Table 6-5 on page 87 for programming guidelines.
29	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
28:23	SYSDIV2	R/W	0x0F	System Clock Divisor 2 Specifies which divisor is used to generate the system clock from either the PLL output or the oscillator source (depending on how the `BYPASS2` bit is configured). `SYSDIV2` is used for the divisor when both the `USESYSDIV` bit in the **RCC** register and the `USERCC2` bit in this register are set. See Table 6-5 on page 87 for programming guidelines.
22	SYSDIV2LSB	R/W	0	Additional LSB for `SYSDIV2` When `DIV400` is set, this bit becomes the LSB of `SYSDIV2`. If `DIV400` is clear, this bit is not used. See Table 6-5 on page 87 for programming guidelines. This bit can only be set or cleared when `DIV400` is set.
21:14	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	PWRDN2	R/W	1	Power-Down PLL 2 Value Description 1 The PLL is powered down. 0 The PLL operates normally.
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	BYPASS2	R/W	1	PLL Bypass 2 Value Description 1 The system clock is derived from the OSC source and divided by the divisor specified by `SYSDIV2`. 0 The system clock is the PLL output clock divided by the divisor specified by `SYSDIV2`. See Table 6-5 on page 87 for programming guidelines. **Note:** The ADC must be clocked from the PLL or directly from a 16-MHz clock source to operate properly.
10:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
6:4	OSCSRC2	R/W	0x1	Oscillator Source 2 Selects the input source for the OSC. The values are: Value / Description 0x0 MOSC Main oscillator 0x1 PIOSC Precision internal oscillator 0x2 PIOSC/4 Precision internal oscillator / 4 0x3 30 kHz 30-kHz internal oscillator 0x4-0x5 Reserved 0x6 4.194304 MHz 4.194304-MHz external oscillator 0x7 32.768 kHz 32.768-kHz external oscillator
3:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

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

Main Oscillator Control (MOSCCTL)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															CVAL
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Value	Description
1	The MOSC monitor circuit is enabled.
0	The MOSC monitor circuit is disabled.

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

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

Deep Sleep Clock Configuration (DSLPCLKCFG)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			DSDIVORIDE						reserved						
Type	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									DSOSCSRC			reserved			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28:23	DSDIVORIDE	R/W	0x0F	Divider Field Override If Deep-Sleep mode is enabled when the PLL is running, the PLL is disabled. This 6-bit field contains a system divider field that overrides the `SYSDIV` field in the **RCC** register or the `SYSDIV2` field in the **RCC2** register during Deep Sleep. This divider is applied to the source selected by the **DSOSCSRC** field. Value Description 0x0 /1 0x1 /2 0x2 /3 0x3 /4 ... ... 0x3F /64
22:7	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
6:4	DSOSCSRC	R/W	0x0	Clock Source Specifies the clock source during Deep-Sleep mode. Value Description 0x0 MOSC Use the main oscillator as the source. **Note:** If the PIOSC is being used as the clock reference for the PLL, the PIOSC is the clock source instead of MOSC in Deep-Sleep mode. If the Hibernation module 4.194304-MHz crystal is being used as the clock reference for the PLL, the 4.194304-MHz crystal is the clock source instead of MOSC in Deep-Sleep mode. 0x1 PIOSC Use the precision internal 16-MHz oscillator as the source. **Note:** If the Hibernation module 4.194304-MHz crystal is being used as the clock reference for the PLL, the 4.194304-MHz crystal is the clock source instead of PIOSC in Deep-Sleep mode. 0x2 Reserved 0x3 30 kHz Use the 30-kHz internal oscillator as the source. 0x4-0x5 Reserved 0x6 4.194304 MHz Use the Hibernation module 4.194304-MHz external crystal clock as the source. 0x7 32.768 kHz Use the Hibernation module 32.768-kHz external oscillator as the source.
3:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 13: Precision Internal Oscillator Calibration (PIOSCCAL), offset 0x150

This register provides the ability to update or recalibrate the precision internal oscillator. Note that a 32.768-kHz oscillator must be used as the Hibernation module clock source for the user to be able to calibrate the PIOSC.

Precision Internal Oscillator Calibration (PIOSCCAL)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved						CAL	UPDATE	reserved	UT						
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	UTEN	R/W	0	Use User Trim Value Value Description 1 The trim value in bits[6:0] of this register are used for any update trim operation. 0 The factory calibration value is used for an update trim operation.
30:10	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	CAL	R/W	0	Start Calibration Value Description 1 Starts a new calibration of the PIOSC. Results are in the **PIOSCSTAT** register. The resulting trim value from the operation is active in the PIOSC after the calibration completes. The result overrides any previous update trim operation whether the calibration passes or fails. 0 No action. This bit is auto-cleared when the calibration finishes.
8	UPDATE	R/W	0	Update Trim Value Description 1 Updates the PIOSC trim value with the `UT` bit or the `DT` bit in the **PIOSCSTAT** register. Used with `UTEN`. 0 No action. This bit is auto-cleared after the update.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
6:0	UT	R/W	0x0	User Trim Value User trim value that can be loaded into the PIOSC. Refer to “Main PLL Frequency Configuration” on page 89 for more information on calibrating the PIOSC.

Register 14: Precision Internal Oscillator Statistics (PIOSCSTAT), offset 0x154

This register provides the user information on the PIOSC calibration. Note that a 32.768-kHz oscillator must be used as the Hibernation module clock source for the user to be able to calibrate the PIOSC.

Precision Internal Oscillator Statistics (PIOSCSTAT)

Base 0x400F.E000
Offset 0x154
Type RO, reset 0x0000.0040

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved						RESULT		reserved	CT						
Type	RO	RO	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	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:23	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22:16	DT	RO	-	Default Trim Value This field contains the default trim value. This value is loaded into the PIOSC after every full power-up.
15:10	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	RESULT	RO	0	Calibration Result Value Description 0x0 Calibration has not been attempted. 0x1 The last calibration operation completed to meet 1% accuracy. 0x2 The last calibration operation failed to meet 1% accuracy. 0x3 Reserved
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	CT	RO	0x40	Calibration Trim Value This field contains the trim value from the last calibration operation. After factory calibration `CT` and `DT` are the same.

Register 15: Device Identification 1 (DID1), offset 0x004

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

Device Identification 1 (DID1)

Base 0x400F.E000
Offset 0x004
Type RO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	VER				FAM				PARTNO							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	1	0	0	0	0	0	0	1	1	0	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	1	0	0	0	0	0	-	-	-	-	-	1	-	-

Bit/Field	Name	Type	Reset	Description
31:28	VER	RO	0x1	DID1 Version This field defines the **DID1** register format version. The version number is numeric. The value of the VER field is encoded as follows (all other encodings are reserved): Value Description 0x1 Second version of the **DID1** register format.
27:24	FAM	RO	0x0	Family This field provides the family identification of the device within the Luminary Micro product portfolio. The value is encoded as follows (all other encodings are reserved): Value Description 0x0 Stellaris family of microcontollers, that is, all devices with external part numbers starting with LM3S.
23:16	PARTNO	RO	0x30	Part Number This field provides the part number of the device within the family. The value is encoded as follows (all other encodings are reserved): Value Description 0x30 LM3S1W16
15:13	PINCOUNT	RO	0x3	Package Pin Count This field specifies the number of pins on the device package. The value is encoded as follows (all other encodings are reserved): Value Description 0x3 64-pin package

Bit/Field	Name	Type	Reset	Description
12:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	TEMP	RO	-	Temperature Range This field specifies the temperature rating of the device. The value is encoded as follows (all other encodings are reserved): Value Description 0x0 Commercial temperature range (0°C to 70°C) 0x1 Industrial temperature range (-40°C to 85°C) 0x2 Extended temperature range (-40°C to 105°C)
4:3	PKG	RO	-	Package Type This field specifies the package type. The value is encoded as follows (all other encodings are reserved): Value Description 0x0 SOIC package 0x1 LQFP package 0x2 BGA package
2	ROHS	RO	1	RoHS-Compliance This bit specifies whether the device is RoHS-compliant. A 1 indicates the part is RoHS-compliant.
1:0	QUAL	RO	-	Qualification Status This field specifies the qualification status of the device. The value is encoded as follows (all other encodings are reserved): Value Description 0x0 Engineering Sample (unqualified) 0x1 Pilot Production (unqualified) 0x2 Fully Qualified

Register 16: Device Capabilities 0 (DC0), offset 0x008

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

Device Capabilities 0 (DC0)

Base 0x400F.E000
Offset 0x008
Type RO, reset 0x001F.000F

	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	0	0	0	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FLASHSZ															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:16	SRAMSZ	RO	0x001F	SRAM Size Indicates the size of the on-chip SRAM memory.
15:0	FLASHSZ	RO	0x000F	Flash Size Indicates the size of the on-chip flash memory.

SRAMSZ values:

Value	Description
0x001F	8 KB of SRAM

FLASHSZ values:

Value	Description
0x000F	32 KB of Flash

Register 17: Device Capabilities 1 (DC1), offset 0x010

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

Device Capabilities 1 (DC1)

Base 0x400F.E000
Offset 0x010
Type RO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			WDT1	reserved											ADC0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	1	0	0	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
	MINSYSDIV				reserved		MAXADC0SPD		MPU	HIB	TEMPSNS	PLL	WDT0	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	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	WDT1	RO	1	Watchdog Timer1 Present When set, indicates that watchdog timer 1 is present.
27:17	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	ADC0	RO	1	ADC Module 0 Present When set, indicates that ADC module 0 is present
15:12	MINSYSDIV	RO	-	System Clock Divider Minimum 4-bit divider value for system clock. The reset value is hardware-dependent. See the RCC register for how to change the system clock divisor using the SYSDIV bit. Value Description 0x1 Divide VCO (400MHZ) by 5 minimum 0x2 Divide VCO (400MHZ) by 2*2 + 2 = 6 minimum 0x3 Specifies a 50-MHz CPU clock with a PLL divider of 4. 0x7 Specifies a 25-MHz clock with a PLL divider of 8. 0x9 Specifies a 20-MHz clock with a PLL divider of 10.
11:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
9:8	MAXADC0SPD	RO	0x3	Max ADC0 Speed This field indicates the maximum rate at which the ADC samples data. Value Description 0x3 1M samples/second
7	MPU	RO	1	MPU Present When set, indicates that the Cortex-M3 Memory Protection Unit (MPU) module is present. See the ARM Cortex-M3 Technical Reference Manual for details on the MPU.
6	HIB	RO	1	Hibernation Module Present When set, indicates that the Hibernation module is present.
5	TEMPSNS	RO	1	Temp Sensor Present When set, indicates that the on-chip temperature sensor is present.
4	PLL	RO	1	PLL Present When set, indicates that the on-chip Phase Locked Loop (PLL) is present.
3	WDT0	RO	1	Watchdog Timer 0 Present When set, indicates that watchdog timer 0 is present.
2	SWO	RO	1	SWO Trace Port Present When set, indicates that the Serial Wire Output (SWO) trace port is present.
1	SWD	RO	1	SWD Present When set, indicates that the Serial Wire Debugger (SWD) is present.
0	JTAG	RO	1	JTAG Present When set, indicates that the JTAG debugger interface is present.

Register 18: Device Capabilities 2 (DC2), offset 0x014

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

Device Capabilities 2 (DC2)

Base 0x400F.E000
Offset 0x014
Type RO, reset 0x0307.5037

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved						COMP1	COMP0	reserved					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	0	1	1	0	0	0	0	0	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	I2C1	reserved	I2C0	reserved						SSI1	SSI0	reserved	UART2	UART1	UART0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	1	0	1	0	0	0	0	0	0	1	1	0	1	1	1

Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	COMP1	RO	1	Analog Comparator 1 Present When set, indicates that analog comparator 1 is present.
24	COMP0	RO	1	Analog Comparator 0 Present When set, indicates that analog comparator 0 is present.
23:19	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
18	TIMER2	RO	1	Timer Module 2 Present When set, indicates that General-Purpose Timer module 2 is present.
17	TIMER1	RO	1	Timer Module 1 Present When set, indicates that General-Purpose Timer module 1 is present.
16	TIMER0	RO	1	Timer Module 0 Present When set, indicates that General-Purpose Timer module 0 is present.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	RO	1	I2C Module 1 Present When set, indicates that I2C module 1 is present.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
12	I2C0	RO	1	I2C Module 0 Present When set, indicates that I2C module 0 is present.
11:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	RO	1	SSI Module 1 Present When set, indicates that SSI module 1 is present.
4	SSI0	RO	1	SSI Module 0 Present When set, indicates that SSI module 0 is present.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	RO	1	UART Module 2 Present When set, indicates that UART module 2 is present.
1	UART1	RO	1	UART Module 1 Present When set, indicates that UART module 1 is present.
0	UART0	RO	1	UART Module 0 Present When set, indicates that UART module 0 is present.

Register 19: Device Capabilities 3 (DC3), offset 0x018

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

Device Capabilities 3 (DC3)

Base 0x400F.E000
Offset 0x018
Type RO, reset 0xBFFF.0FC0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	32KHZ	reserved	CCP5	CCP4	CCP3	CCP2	CCP1	CCP0	ADC0AIN7	ADC0AIN6	ADC0AIN5	ADC0AIN4	ADC0AIN3	ADC0AIN2	ADC0AIN1	ADC0AIN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				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	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
21	ADC0AIN5	RO	1	ADC Module 0 AIN5 Pin Present When set, indicates that ADC module 0 input pin 5 is present.
20	ADC0AIN4	RO	1	ADC Module 0 AIN4 Pin Present When set, indicates that ADC module 0 input pin 4 is present.
19	ADC0AIN3	RO	1	ADC Module 0 AIN3 Pin Present When set, indicates that ADC module 0 input pin 3 is present.
18	ADC0AIN2	RO	1	ADC Module 0 AIN2 Pin Present When set, indicates that ADC module 0 input pin 2 is present.
17	ADC0AIN1	RO	1	ADC Module 0 AIN1 Pin Present When set, indicates that ADC module 0 input pin 1 is present.
16	ADC0AIN0	RO	1	ADC Module 0 AIN0 Pin Present When set, indicates that ADC module 0 input pin 0 is present.
15:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	C1O	RO	1	C1o Pin Present When set, indicates that the analog comparator 1 output pin is present.
10	C1PLUS	RO	1	C1+ Pin Present When set, indicates that the analog comparator 1 (+) input pin is present.
9	C1MINUS	RO	1	C1- Pin Present When set, indicates that the analog comparator 1 (-) input pin is present.
8	C0O	RO	1	C0o Pin Present When set, indicates that the analog comparator 0 output pin is present.
7	C0PLUS	RO	1	C0+ Pin Present When set, indicates that the analog comparator 0 (+) input pin is present.
6	C0MINUS	RO	1	C0- Pin Present When set, indicates that the analog comparator 0 (-) input pin is present.
5:0	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: Device Capabilities 4 (DC4), offset 0x01C

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

Device Capabilities 4 (DC4)

Base 0x400F.E000
Offset 0x01C
Type RO, reset 0x0004.301F

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved													PICAL	reserved	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		UDMA	ROM	reserved							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	1	1	0	0	0	0	0	0	0	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:19	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
18	PICAL	RO	1	PIOSC Calibrate When set, indicates that the PIOSC can be calibrated by software.
17:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	RO	1	Micro-DMA Module Present When set, indicates that the micro-DMA module present.
12	ROM	RO	1	Internal Code ROM Present When set, indicates that internal code ROM is present.
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	GPIOE	RO	1	GPIO Port E Present When set, indicates that GPIO Port E is present.
3	GPIOD	RO	1	GPIO Port D Present When set, indicates that GPIO Port D is present.
2	GPIOC	RO	1	GPIO Port C Present When set, indicates that GPIO Port C is present.
1	GPIOB	RO	1	GPIO Port B Present When set, indicates that GPIO Port B is present.

Bit/Field	Name	Type	Reset	Description
0	GPIOA	RO	1	GPIO Port A Present When set, indicates that GPIO Port A is present.

Register 21: Device Capabilities 5 (DC5), offset 0x020

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

Device Capabilities 5 (DC5)

Base 0x400F.E000
Offset 0x020
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 22: Device Capabilities 6 (DC6), offset 0x024

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

Device Capabilities 6 (DC6)

Base 0x400F.E000
Offset 0x024
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 23: Device Capabilities 7 (DC7), offset 0x028

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

Device Capabilities 7 (DC7)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	DMACH30	DMACH29	DMACH28	DMACH27	DMACH26	DMACH25	DMACH24	DMACH23	DMACH22	DMACH21	DMACH20	DMACH19	DMACH18	DMACH17	DMACH16
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DMACH15	DMACH14	DMACH13	DMACH12	DMACH11	DMACH10	DMACH9	DMACH8	DMACH7	DMACH6	DMACH5	DMACH4	DMACH3	DMACH2	DMACH1	DMACH0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

Bit/Field	Name	Type	Reset	Description
25	DMACH25	RO	1	SSI1_TX / ADC1_SS1 When set, indicates uDMA channel 25 is available and connected to the transmit path of SSI module 1. If the corresponding bit in the **DMACHALT** register is set, the channel is connected instead to the alternate channel assignment of ADC module 1 Sample Sequencer 1.
24	DMACH24	RO	1	SSI1_RX / ADC1_SS0 When set, indicates uDMA channel 24 is available and connected to the receive path of SSI module 1. If the corresponding bit in the **DMACHALT** register is set, the channel is connected instead to the alternate channel assignment of ADC module 1 Sample Sequencer 0.
23	DMACH23	RO	1	UART1_TX / CAN2_TX When set, indicates uDMA channel 23 is available and connected to the transmit path of UART module 1. If the corresponding bit in the **DMACHALT** register is set, the channel is connected instead to the alternate channel assignment of CAN module 2 transmit.
22	DMACH22	RO	1	UART1_RX / CAN2_RX When set, indicates uDMA channel 22 is available and connected to the receive path of UART module 1. If the corresponding bit in the **DMACHALT** register is set, the channel is connected instead to the alternate channel assignment of CAN module 2 receive.
21	DMACH21	RO	1	Timer1B / EPI0_WFIFO When set, indicates uDMA channel 21 is available and connected to Timer 1B.
20	DMACH20	RO	1	Timer1A / EPI0_NBRFIFO When set, indicates uDMA channel 20 is available and connected to Timer 1A.
19	DMACH19	RO	1	Timer0B / Timer1B When set, indicates uDMA channel 19 is available and connected to Timer 0B. If the corresponding bit in the **DMACHALT** register is set, the channel is connected instead to the alternate channel assignment of Timer 1B.
18	DMACH18	RO	1	Timer0A / Timer1A When set, indicates uDMA channel 18 is available and connected to Timer 0A. If the corresponding bit in the **DMACHALT** register is set, the channel is connected instead to the alternate channel assignment of Timer 1A.
17	DMACH17	RO	1	ADC0_SS3 When set, indicates uDMA channel 17 is available and connected to ADC module 0 Sample Sequencer 3.
16	DMACH16	RO	1	ADC0_SS2 When set, indicates uDMA channel 16 is available and connected to ADC module 0 Sample Sequencer 2.

Bit/Field	Name	Type	Reset	Description
15	DMACH15	RO	1	ADC0_SS1 / Timer2B When set, indicates uDMA channel 15 is available and connected to ADC module 0 Sample Sequencer 1. If the corresponding bit in the **DMACHALT** register is set, the channel is connected instead to the alternate channel assignment of Timer 2B.
14	DMACH14	RO	1	ADC0_SS0 / Timer2A When set, indicates uDMA channel 14 is available and connected to ADC module 0 Sample Sequencer 0. If the corresponding bit in the **DMACHALT** register is set, the channel is connected instead to the alternate channel assignment of Timer 2A.
13	DMACH13	RO	1	CAN0_TX / UART2_TX When set, indicates uDMA channel 13 is available and connected to the transmit path of CAN module 0. If the corresponding bit in the **DMACHALT** register is set, the channel is connected instead to the alternate channel assignment of UART module 2 transmit.
12	DMACH12	RO	1	CAN0_RX / UART2_RX When set, indicates uDMA channel 12 is available and connected to the receive path of CAN module 0. If the corresponding bit in the **DMACHALT** register is set, the channel is connected instead to the alternate channel assignment of UART module 2 receive.
11	DMACH11	RO	1	SSI0_TX / SSI1_TX When set, indicates uDMA channel 11 is available and connected to the transmit path of SSI module 0. If the corresponding bit in the **DMACHALT** register is set, the channel is connected instead to the alternate channel assignment of SSI module 1 transmit.
10	DMACH10	RO	1	SSI0_RX / SSI1_RX When set, indicates uDMA channel 10 is available and connected to the receive path of SSI module 0. If the corresponding bit in the **DMACHALT** register is set, the channel is connected instead to the alternate channel assignment of SSI module 1 receive.
9	DMACH9	RO	1	UART0_TX / UART1_TX When set, indicates uDMA channel 9 is available and connected to the transmit path of UART module 0. If the corresponding bit in the **DMACHALT** register is set, the channel is connected instead to the alternate channel assignment of UART module 1 transmit.
8	DMACH8	RO	1	UART0_RX / UART1_RX When set, indicates uDMA channel 8 is available and connected to the receive path of UART module 0. If the corresponding bit in the **DMACHALT** register is set, the channel is connected instead to the alternate channel assignment of UART module 1 receive.
7	DMACH7	RO	1	ETH_TX / Timer2B When set, indicates uDMA channel 7 is available and connected to the transmit path of the Ethernet module. If the corresponding bit in the **DMACHALT** register is set, the channel is connected instead to the alternate channel assignment of Timer 2B.

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

Register 24: Device Capabilities 8 ADC Channels (DC8), offset 0x02C

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

Device Capabilities 8 ADC Channels (DC8)

Base 0x400F.E000
Offset 0x02C
Type RO, reset 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								ADC0AIN7	ADC0AIN6	ADC0AIN5	ADC0AIN4	ADC0AIN3	ADC0AIN2	ADC0AIN1	ADC0AIN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	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	ADC0AIN7	RO	1	ADC Module 0 AIN7 Pin Present When set, indicates that ADC module 0 input pin 7 is present.
6	ADC0AIN6	RO	1	ADC Module 0 AIN6 Pin Present When set, indicates that ADC module 0 input pin 6 is present.
5	ADC0AIN5	RO	1	ADC Module 0 AIN5 Pin Present When set, indicates that ADC module 0 input pin 5 is present.
4	ADC0AIN4	RO	1	ADC Module 0 AIN4 Pin Present When set, indicates that ADC module 0 input pin 4 is present.
3	ADC0AIN3	RO	1	ADC Module 0 AIN3 Pin Present When set, indicates that ADC module 0 input pin 3 is present.
2	ADC0AIN2	RO	1	ADC Module 0 AIN2 Pin Present When set, indicates that ADC module 0 input pin 2 is present.
1	ADC0AIN1	RO	1	ADC Module 0 AIN1 Pin Present When set, indicates that ADC module 0 input pin 1 is present.
0	ADC0AIN0	RO	1	ADC Module 0 AIN0 Pin Present When set, indicates that ADC module 0 input pin 0 is present.

Register 25: Device Capabilities 9 ADC Digital Comparators (DC9), offset 0x190

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

Device Capabilities 9 ADC Digital Comparators (DC9)

Base 0x400F.E000
Offset 0x190
Type RO, reset 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								ADC0DC7	ADC0DC6	ADC0DC5	ADC0DC4	ADC0DC3	ADC0DC2	ADC0DC1	ADC0DC0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	ADC0DC7	RO	1	ADC0 DC7 Present When set, indicates that ADC module 0 Digital Comparator 7 is present.
6	ADC0DC6	RO	1	ADC0 DC6 Present When set, indicates that ADC module 0 Digital Comparator 6 is present.
5	ADC0DC5	RO	1	ADC0 DC5 Present When set, indicates that ADC module 0 Digital Comparator 5 is present.
4	ADC0DC4	RO	1	ADC0 DC4 Present When set, indicates that ADC module 0 Digital Comparator 4 is present.
3	ADC0DC3	RO	1	ADC0 DC3 Present When set, indicates that ADC module 0 Digital Comparator 3 is present.
2	ADC0DC2	RO	1	ADC0 DC2 Present When set, indicates that ADC module 0 Digital Comparator 2 is present.
1	ADC0DC1	RO	1	ADC0 DC1 Present When set, indicates that ADC module 0 Digital Comparator 1 is present.
0	ADC0DC0	RO	1	ADC0 DC0 Present When set, indicates that ADC module 0 Digital Comparator 0 is present.

Register 26: Non-Volatile Memory Information (NVMSTAT), offset 0x1A0

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

Non-Volatile Memory Information (NVMSTAT)

Base 0x400F.E000
Offset 0x1A0
Type RO, reset 0x0000.0001

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															FWB
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

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

Register 27: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100

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

Run Mode Clock Gating Control Register 0 (RCGC0)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			WDT1	reserved											ADC0
Type	RO	RO	RO	R/W	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved						MAXADC0SPD		reserved	HIB	reserved		WDT0	reserved		
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	R/W	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	WDT1	R/W	0	WDT1 Clock Gating Control This bit controls the clock gating for the Watchdog Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
27:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	ADC0	R/W	0	ADC0 Clock Gating Control This bit controls the clock gating for ADC module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
9:8	MAXADC0SPD	R/W	0	ADC0 Sample Speed This field sets the rate at which ADC0 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the `MAXADC0SPD` bit as follows (all other encodings are reserved): 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	1	HIB Clock Gating Control This bit controls the clock gating for the Hibernation module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT0	R/W	0	WDT0 Clock Gating Control This bit controls the clock gating for the Watchdog Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 28: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110

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

Sleep Mode Clock Gating Control Register 0 (SCGC0)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			WDT1	reserved											ADC0
Type	RO	RO	RO	R/W	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved						MAXADC0SPD		reserved	HIB	reserved		WDT0	reserved		
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	R/W	RO	RO	R/W	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	WDT1	R/W	0	WDT1 Clock Gating Control This bit controls the clock gating for Watchdog Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
27:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	ADC0	R/W	0	ADC0 Clock Gating Control This bit controls the clock gating for ADC module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
9:8	MAXADC0SPD	R/W	0	ADC0 Sample Speed This field sets the rate at which ADC module 0 samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the `MAXADC0SPD` bit as follows (all other encodings are reserved): 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	1	HIB Clock Gating Control This bit controls the clock gating for the Hibernation module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT0	R/W	0	WDT0 Clock Gating Control This bit controls the clock gating for the Watchdog Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 29: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120

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

Deep Sleep Mode Clock Gating Control Register 0 (DCGC0)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			WDT1	reserved											ADC0
Type	RO	RO	RO	R/W	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									HIB	reserved		WDT0	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	1	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	WDT1	R/W	0	WDT1 Clock Gating Control This bit controls the clock gating for the Watchdog Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
27:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	ADC0	R/W	0	ADC0 Clock Gating Control This bit controls the clock gating for ADC module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
6	HIB	R/W	1	HIB Clock Gating Control This bit controls the clock gating for the Hibernation module. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT0	R/W	0	WDT0 Clock Gating Control This bit controls the clock gating for the Watchdog Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 30: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104

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

Run Mode Clock Gating Control Register 1 (RCGC1)

Base 0x400F.E000
Offset 0x104
Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved						COMP1	COMP0	reserved					TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	I2C1	reserved	I2C0	reserved						SSI1	SSI0	reserved	UART2	UART1	UART0
Type	RO	R/W	RO	R/W	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	COMP1	R/W	0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
23:19	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
17	TIMER1	R/W	0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	I2C1 Clock Gating Control This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Clock Gating Control This bit controls the clock gating for I2C module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
11:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	R/W	0	SSI1 Clock Gating Control This bit controls the clock gating for SSI module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
4	SSI0	R/W	0	SSI0 Clock Gating Control This bit controls the clock gating for SSI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
2	UART2	R/W	0	UART2 Clock Gating Control This bit controls the clock gating for UART module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	UART1	R/W	0	UART1 Clock Gating Control This bit controls the clock gating for UART module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control This bit controls the clock gating for UART module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Register 31: Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114

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

Sleep Mode Clock Gating Control Register 1 (SCGC1)

Base 0x400F.E000
Offset 0x114
Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved						COMP1	COMP0	reserved					TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	I2C1	reserved	I2C0	reserved						SSI1	SSI0	reserved	UART2	UART1	UART0
Type	RO	R/W	RO	R/W	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	COMP1	R/W	0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
23:19	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
17	TIMER1	R/W	0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	I2C1 Clock Gating Control This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Clock Gating Control This bit controls the clock gating for I2C module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
11:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	R/W	0	SSI1 Clock Gating Control This bit controls the clock gating for SSI module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
4	SSI0	R/W	0	SSI0 Clock Gating Control This bit controls the clock gating for SSI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
2	UART2	R/W	0	UART2 Clock Gating Control This bit controls the clock gating for UART module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	UART1	R/W	0	UART1 Clock Gating Control This bit controls the clock gating for UART module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control This bit controls the clock gating for UART module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Register 32: Deep-Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124

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

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

Base 0x400F.E000
Offset 0x124
Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved						COMP1	COMP0	reserved					TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	I2C1	reserved	I2C0	reserved						SSI1	SSI0	reserved	UART2	UART1	UART0
Type	RO	R/W	RO	R/W	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	COMP1	R/W	0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
23:19	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
18	TIMER2	R/W	0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
17	TIMER1	R/W	0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	I2C1 Clock Gating Control This bit controls the clock gating for I2C module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Clock Gating Control This bit controls the clock gating for I2C module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
11:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	R/W	0	SSI1 Clock Gating Control This bit controls the clock gating for SSI module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
4	SSI0	R/W	0	SSI0 Clock Gating Control This bit controls the clock gating for SSI module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
2	UART2	R/W	0	UART2 Clock Gating Control This bit controls the clock gating for UART module 2. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	UART1	R/W	0	UART1 Clock Gating Control This bit controls the clock gating for UART module 1. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control This bit controls the clock gating for UART module 0. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Register 33: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108

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

Run Mode Clock Gating Control Register 2 (RCGC2)

Base 0x400F.E000
Offset 0x108
Type R/W, reset 0x00000000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		UDMA	reserved								GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	R/W	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
31:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	R/W	0	Micro-DMA Clock Gating Control This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
12:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	GPIOE	R/W	0	Port E Clock Gating Control Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	GPIOD	R/W	0	Port D Clock Gating Control Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
2	GPIOC	R/W	0	Port C Clock Gating Control This bit controls the clock gating for Port C. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	GPIOB	R/W	0	Port B Clock Gating Control This bit controls the clock gating for Port B. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	GPIOA	R/W	0	Port A Clock Gating Control This bit controls the clock gating for Port A. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Register 34: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118

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

Sleep Mode Clock Gating Control Register 2 (SCGC2)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		UDMA	reserved								GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	R/W	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
31:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	R/W	0	Micro-DMA Clock Gating Control This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
12:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	GPIOE	R/W	0	Port E Clock Gating Control Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	GPIOD	R/W	0	Port D Clock Gating Control Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
2	GPIOC	R/W	0	Port C Clock Gating Control This bit controls the clock gating for Port C. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	GPIOB	R/W	0	Port B Clock Gating Control This bit controls the clock gating for Port B. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	GPIOA	R/W	0	Port A Clock Gating Control This bit controls the clock gating for Port A. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Register 35: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128

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

Deep Sleep Mode Clock Gating Control Register 2 (DCGC2)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		UDMA	reserved								GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	R/W	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
31:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	R/W	0	Micro-DMA Clock Gating Control This bit controls the clock gating for micro-DMA. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
12:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	GPIOE	R/W	0	Port E Clock Gating Control Port E Clock Gating Control. This bit controls the clock gating for Port E. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
3	GPIOD	R/W	0	Port D Clock Gating Control Port D Clock Gating Control. This bit controls the clock gating for Port D. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Bit/Field	Name	Type	Reset	Description
2	GPIOC	R/W	0	Port C Clock Gating Control This bit controls the clock gating for Port C. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
1	GPIOB	R/W	0	Port B Clock Gating Control This bit controls the clock gating for Port B. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.
0	GPIOA	R/W	0	Port A Clock Gating Control This bit controls the clock gating for Port A. If set, the module receives a clock and functions. Otherwise, the module is unclocked and disabled. If the module is unclocked, a read or write to the module generates a bus fault.

Register 36: Software Reset Control 0 (SRCR0), offset 0x040

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

Software Reset Control 0 (SRCR0)

Base 0x400F.E000
Offset 0x040
Type R/W, reset 0x00000000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			WDT1	reserved											ADC0
Type	RO	RO	RO	R/W	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									HIB	reserved		WDT0	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
31:29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	WDT1	R/W	0	WDT1 Reset Control When this bit is set, Watchdog Timer module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
27:17	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	ADC0	R/W	0	ADC0 Reset Control When this bit is set, ADC module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
15:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	HIB	R/W	0	HIB Reset Control When this bit is set, the Hibernation module is reset. All internal data is lost and the registers are returned to their reset states.This bit must be manually cleared after being set.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT0	R/W	0	WDT0 Reset Control When this bit is set, Watchdog Timer module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.

Bit/Field	Name	Type	Reset	Description
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 37: Software Reset Control 1 (SRCR1), offset 0x044

This register allows individual modules to be reset. Writes to this register are masked by the bits in the **Device Capabilities 2 (DC2)** register.

Software Reset Control 1 (SRCR1)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved						COMP1	COMP0	reserved					TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	I2C1	reserved	I2C0	reserved						SSI1	SSI0	reserved	UART2	UART1	UART0
Type	RO	R/W	RO	R/W	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	COMP1	R/W	0	Analog Comp 1 Reset Control When this bit is set, Analog Comparator module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
24	COMP0	R/W	0	Analog Comp 0 Reset Control When this bit is set, Analog Comparator module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
23:19	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
18	TIMER2	R/W	0	Timer 2 Reset Control When this bit is set, General-Purpose Timer module 2 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
17	TIMER1	R/W	0	Timer 1 Reset Control When this bit is set, General-Purpose Timer module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
16	TIMER0	R/W	0	Timer 0 Reset Control When this bit is set, General-Purpose Timer module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
14	I2C1	R/W	0	I2C1 Reset Control When this bit is set, I2C module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Reset Control When this bit is set, I2C module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
11:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	R/W	0	SSI1 Reset Control When this bit is set, SSI module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
4	SSI0	R/W	0	SSI0 Reset Control When this bit is set, SSI module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	UART2	R/W	0	UART2 Reset Control When this bit is set, UART module 2 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
1	UART1	R/W	0	UART1 Reset Control When this bit is set, UART module 1 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
0	UART0	R/W	0	UART0 Reset Control When this bit is set, UART module 0 is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.

Register 38: Software Reset Control 2 (SRCR2), offset 0x048

This register allows individual modules to be reset. Writes to this register are masked by the bits in the **Device Capabilities 4 (DC4)** register.

Software Reset Control 2 (SRCR2)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		UDMA	reserved								GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	R/W	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
31:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	R/W	0	Micro-DMA Reset Control When this bit is set, uDMA module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
12:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	GPIOE	R/W	0	Port E Reset Control When this bit is set, Port E module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
3	GPIOD	R/W	0	Port D Reset Control When this bit is set, Port D module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
2	GPIOC	R/W	0	Port C Reset Control When this bit is set, Port C module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.
1	GPIOB	R/W	0	Port B Reset Control When this bit is set, Port B module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.

Bit/Field	Name	Type	Reset	Description
0	GPIOA	R/W	0	Port A Reset Control When this bit is set, Port A module is reset. All internal data is lost and the registers are returned to their reset states. This bit must be manually cleared after being set.

7 Hibernation Module

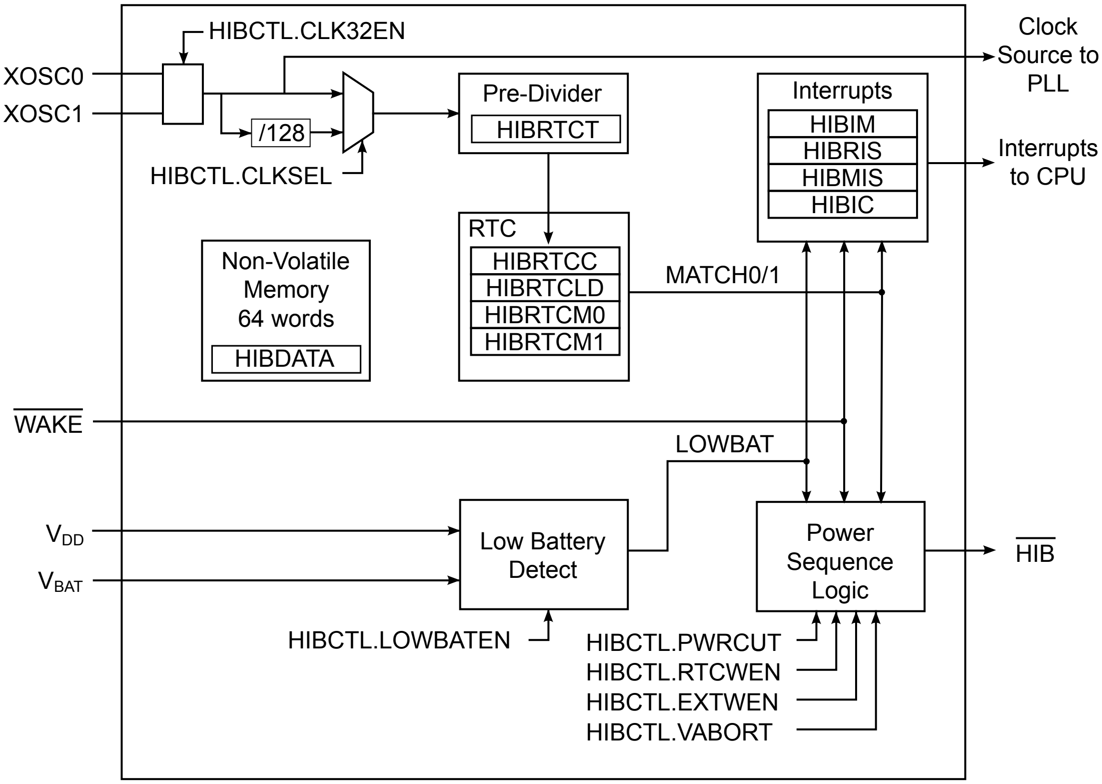
The Hibernation Module manages removal and restoration of power to provide a means for reducing power consumption. When the processor and peripherals are idle, power can be completely removed with only the Hibernation module remaining powered. Power can be restored based on an external signal or at a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

The Hibernation module has the following features:

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

7.1 Block Diagram

Figure 7-1. Hibernation Module Block Diagram



7.2 Signal Description

Table 7-1 on page 169 lists the external signals of the Hibernation module and describes the function of each. These signals have dedicated functions and are not alternate functions for any GPIO signals.

Table 7-1. Signals for Hibernate

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
HIB	33	fixed	O	OD	An open-drain output that indicates the processor is in Hibernate mode.
VBAT	37	fixed	-	Power	Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply.
WAKE	32	fixed	I	TTL	An external input that brings the processor out of Hibernate mode when asserted.
XOSC0	34	fixed	I	Analog	Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the `CLKSEL` bit in the **HIBCTL** register.
XOSC1	35	fixed	O	Analog	Hibernation module oscillator crystal output.

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

7.3 Functional Description

Important: The Hibernate module must have either the RTC function or the External Wake function enabled to ensure proper operation of the microcontroller. See “Initialization” on page 174.

The Hibernation module provides two mechanisms for power control:

- The first mechanism controls the power to the microcontroller with a control signal ($\overline{\texttt{HIB}}$) that signals an external voltage regulator to turn on or off.
- The second mechanism uses internal switches to control power to the Cortex-M3 as well as to most analog and digital functions while retaining I/O pin power (VDD3ON mode).

The Hibernation module power source is determined dynamically. The supply voltage of the Hibernation module is the larger of the main voltage source (V_{DD}) or the battery/auxilliary voltage source (V_{BAT}). Care must be taken that the voltage amplitude of the 32-kHz oscillator is less than V_{BAT}, otherwise, the Hibernation module draws power from the oscillator and not V_{BAT}. The Hibernation module also has a separate clock source to maintain a real-time clock (RTC). Once in hibernation, the module signals an external voltage regulator to turn back on the power when an external pin ($\overline{\texttt{WAKE}}$) is asserted or when the internal RTC reaches a certain value. The Hibernation module can also detect when the battery voltage is low and optionally prevent hibernation when this occurs.

Power-up from a power cut to code execution is defined as the regulator turn-on time (specified at $t_{HIB\_TO\_VDD}$ maximum) plus the normal chip POR (see “Hibernation Module” on page 652).

7.3.1 Register Access Timing

Because the Hibernation module has an independent clocking domain, certain registers must be written only with a timing gap between accesses. The delay time is $t_{HIB\_REG\_WRITE}$, therefore software must guarantee that this delay is inserted between back-to-back writes to certain Hibernation registers or between a write followed by a read to those same registers. The timing for back-to-back reads from the Hibernation module has no restrictions. Software may make use of the `WRC` bit in the **Hibernation Control (HIBCTL)** register to ensure that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **HIBCTL** for `WRC`=1 prior to accessing any affected register. The following registers are subject to this timing restriction:

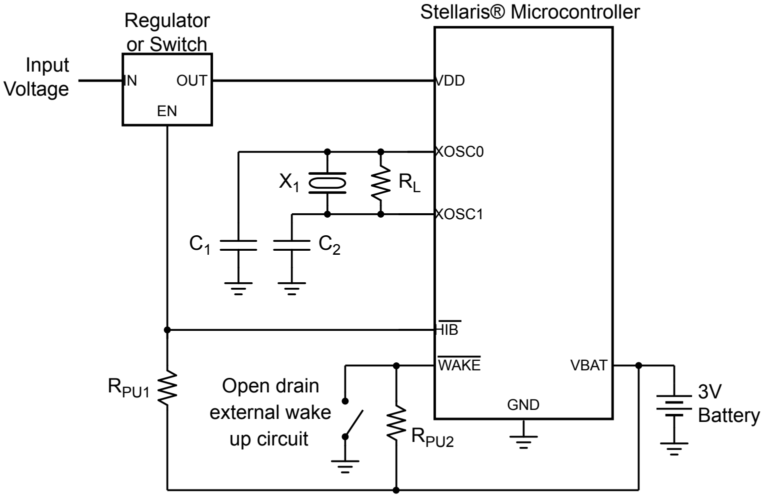
- **Hibernation RTC Counter (HIBRTCC)**
- **Hibernation RTC Match 0 (HIBRTCM0)**
- **Hibernation RTC Match 1 (HIBRTCM1)**
- **Hibernation RTC Load (HIBRTCLD)**
- **Hibernation RTC Trim (HIBRTCT)**
- **Hibernation Data (HIBDATA)**

7.3.2 Clock Source

The Hibernation module must be clocked by an external source, even if the RTC feature is not used. An external oscillator or crystal can be used for this purpose. To use a crystal, a 4.194304-MHz crystal is connected to the `XOSC0` and `XOSC1` pins. This clock signal is divided by 128 internally to produce the 32.768-kHz clock reference. For an alternate clock source, a 32.768-kHz oscillator can be connected to the `XOSC0` pin. Care must be taken that the voltage amplitude of the 32-kHz oscillator is less than V_{BAT}, otherwise, the Hibernation module draws power from the oscillator and not V_{BAT} during hibernation. See Figure 7-2 on page 171 and Figure 7-3 on page 172. Note that these diagrams only show the connection to the Hibernation pins and not to the full system. See "Hibernation Module" on page 652 for specific values.

The clock source is enabled by setting the `CLK32EN` bit of the **HIBCTL** register. The type of clock source is selected by clearing the `CLKSEL` bit for a 4.194304-MHz clock source and setting the `CLKSEL` bit for 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 writing to the `CLK32EN` bit and before any other accesses to the Hibernation module registers. The delay allows the crystal to power up and stabilize. If an oscillator is used for the clock source, no delay is needed.

Figure 7-2. Clock Source Using Crystal



Note: X_1 = Crystal frequency is f_{XOSC_XTAL}.

$C_{1,2}$ = Capacitor value derived from crystal vendor load capacitance specifications.

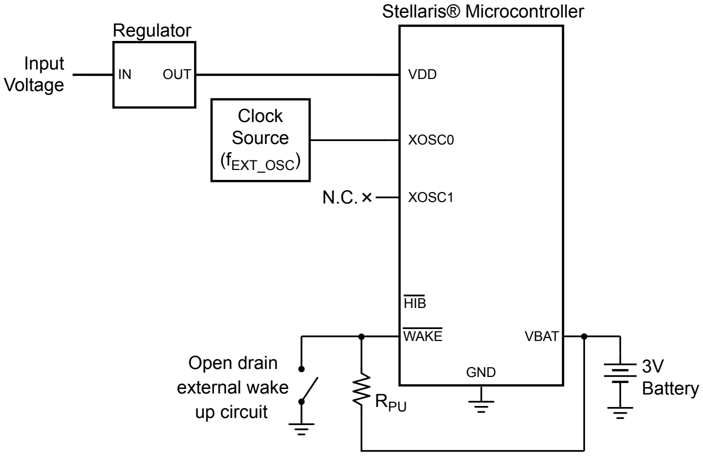
R_L = Load resistor is R_{XOSC_LOAD}.

R_{PU1} = Pull-up resistor 1 (value and voltage source (V_{BAT} or Input Voltage) determined by regulator or switch enable input characteristics).

R_{PU2} = Pull-up resistor 2 is 1 MΩ

See "Hibernation Module" on page 652 for specific parameter values.

Figure 7-3. Clock Source Using Dedicated Oscillator and VDD3ON Mode



Note: R_{PU} = Pull-up resistor is 1 MΩ

If the application does not require the use of the Hibernation module, the `XOSC0` and `XOSC1` can remain unconnected. In this situation, the Hibernation module registers are not accessible.

7.3.3 Battery Management

The Hibernation module can be independently powered by a battery or an auxiliary power source. The module can monitor the voltage level of the battery and detect when the voltage drops below V_{LOWBAT}. When this happens, an interrupt can be generated. The module can also be configured so that it does not go into Hibernate mode if the battery voltage drops below this threshold. Battery voltage is not measured while in Hibernate mode.

Important: System level factors may affect the accuracy of the low battery detect circuit. The designer should consider battery type, discharge characteristics, and a test load during battery voltage measurements.

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

The Hibernation module can be configured to detect a low battery condition by setting the `LOWBATEN` bit of the **HIBCTL** register. In this configuration, the `LOWBAT` bit of the **Hibernation Raw Interrupt Status (HIBRIS)** register is set when the battery level is low. If the `VABORT` bit in the **HIBCTL** register is also set, then the module is prevented from entering Hibernation mode when a low battery is detected. The module can also be configured to generate an interrupt for the low-battery condition (see “Interrupts and Status” on page 174).

7.3.4 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 170). The 32.768-kHz clock signal is fed into a predivider register that counts down the 32.768-kHz clock ticks to achieve a once per second clock rate for the RTC. The rate can be adjusted to compensate for inaccuracies in the clock source by using the predivider trim register, **HIBRTCT**. This register has a nominal value of 0x7FFF, and is used for one second out of every 64 seconds to divide the input clock. This configuration allows the software to make fine corrections to the clock rate by adjusting the predivider trim register up or down from 0x7FFF. The predivider trim should be adjusted up from 0x7FFF in order to slow down the RTC rate and down from 0x7FFF in order to speed up the RTC rate.

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

The RTC must be enabled with the `RTCEN` bit of the **HIBCTL** register. The value of the RTC can be set at any time by writing to the **HIBRTCLD** register. The predivider trim can be adjusted by reading and writing the **HIBRTCT** register. The predivider uses this register once every 64 seconds to adjust the clock rate. The two match registers can be set by writing to the **HIBRTCM0** and **HIBRTCM1** registers. The RTC can be configured to generate interrupts by using the interrupt registers (see “Interrupts and Status” on page 174).

7.3.5 Non-Volatile Memory

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

7.3.6 Power Control Using $\overline{\text{HIB}}$

Important: The Hibernation Module requires special system implementation considerations when using $\overline{\texttt{HIB}}$ to control power, as it is intended to power-down all other sections of the microcontroller. All system signals and power supplies that connect to the chip must be driven to 0 V_{DC} or powered down with the same regulator controlled by $\overline{\texttt{HIB}}$. See “Hibernation Module” on page 652 for more details.

The Hibernation module controls power to the microcontroller through the use of the $\overline{\texttt{HIB}}$ pin which is intended to be connected to the enable signal of the external regulator(s) providing 3.3 V to the microcontroller and other circuits. When the HIB signal is asserted by the Hibernation module, the external regulator is turned off and no longer powers the microcontroller and any parts of the system that are powered by the regulator. The Hibernation module remains powered from the V_{BAT} supply (which could be a battery or an auxiliary power source) until a Wake event. Power to the microcontroller is restored by deasserting the $\overline{\texttt{HIB}}$ signal, which causes the external regulator to turn power back on to the chip.

7.3.7 Power Control Using VDD3ON Mode

The Hibernation module may also be configured to cut power to all internal modules. In the VDD3ON mode, the regulator should maintain 3.3 V power to the microcontroller during Hibernate. This power control mode is enabled by setting the `VDD3ON` bit in **HIBCTL**.

7.3.8 Initiating Hibernate

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 $\overline{\texttt{WAKE}}$ pin, or by using an RTC match. If a Flash memory write operation is in progress, an interlock feature holds off the transition into Hibernation mode until the write has completed.

The Hibernation module is configured to wake from the external $\overline{\texttt{WAKE}}$ pin by setting the `PINWEN` bit of the **HIBCTL** register. It is configured to wake from RTC match by setting the `RTCWEN` bit. Either one or both of these bits must be set prior to going into hibernation. Note that the WAKE pin uses the Hibernation module's internal power supply as the logic 1 reference.

Upon either external wake-up or RTC match, the Hibernation module delays coming out of hibernation until V_{DD} is above the minimum specified voltage, see Table 21-2 on page 643.

When the Hibernation module wakes, the microcontroller performs a normal power-on reset. Software can detect that the power-on was due to a wake from hibernation by examining the raw interrupt status register (see “Interrupts and Status” on page 174) and by looking for state data in the non-volatile memory (see “Non-Volatile Memory” on page 173).

7.3.9 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 Hibernate module can only generate a single interrupt request to the controller at any given time. The software interrupt handler can service multiple interrupt events by reading the **Hibernation Masked Interrupt Status (HIBMIS)** register. Software can also read the status of the Hibernation module at any time by reading the **HIBRIS** register which shows all of the pending events. This register can be used at power-on to see if a wake condition is pending, which indicates to the software that a hibernation wake occurred.

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

7.4 Initialization and Configuration

The Hibernation module has several different configurations. The following sections show the recommended programming sequence for various scenarios. The examples below assume that a 32.768-kHz oscillator is used, and thus always set the `CLKSEL` bit of the **HIBCTL** register. If a 4.194304-MHz crystal is used instead, then the `CLKSEL` bit remains cleared. Because the Hibernation module runs at 32.768 kHz and is asynchronous to the rest of the system, software must allow a delay of $t_{HIB\_REG\_WRITE}$ after writes to certain registers (see “Register Access Timing” on page 170). The registers that require a delay are listed in a note in “Register Map” on page 177 as well as in each register description.

7.4.1 Initialization

The Hibernation module comes out of reset with the system clock enabled to the module, but if the system clock to the module has been disabled, then it must be re-enabled, even if the RTC feature is not used. See page 141.

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_{HIBOSC_SETTLE} for the crystal to power up and stabilize before performing any other operations with the Hibernation module.

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

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

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

Table 7-2 on page 175 illustrates how the clocks function with various bit setting both in normal operation and in hibernation.

Table 7-2. Hibernation Module Clock Operation

CLK32EN	PINWEN	RTCWEN	CLKSEL	RTCEN	Result Normal Operation	Result Hibernation
0	X	X	X	X	Hibernation module disabled	Hibernation module disabled
1	0	0	0	1	RTC match capability enabled. Module clocked from 4.184304-MHz crystal.	No hibernation
1	0	0	1	1	RTC match capability enabled. Module clocked from 32.768-kHz oscillator.	No hibernation
1	0	1	X	1	Module clocked from selected source	RTC match for wake-up event
1	1	0	X	0	Module clocked from selected source	Clock is powered down during hibernation and powered up again on external wake-up event.
1	1	0	X	1	Module clocked from selected source	Clock is powered up during hibernation for RTC. Wake up on external event.
1	1	1	X	1	Module clocked from selected source	RTC match or external wake-up event, whichever occurs first.

7.4.2 RTC Match Functionality (No Hibernation)

Use the following steps to implement the RTC match functionality of the Hibernation module:

1. Write the required RTC match value to one of the **HIBRTCMn** registers at offset 0x004 or 0x008.
2. Write the required RTC load value to the **HIBRTCLD** register at offset 0x00C.
3. Set the required RTC match interrupt mask in the `RTCALT0` and `RTCALT1` bits (bits 1:0) in the **HIBIM** register at offset 0x014.
4. Write 0x0000.0041 to the **HIBCTL** register at offset 0x010 to enable the RTC to begin counting.

7.4.3 RTC Match/Wake-Up from Hibernation

Use the following steps to implement the RTC match and wake-up functionality of the Hibernation module:

1. Write the required RTC match value to the **HIBRTCMn** registers at offset 0x004 or 0x008.
2. Write the required RTC load value to the **HIBRTCLD** register at offset 0x00C.
3. Write any data to be retained during power cut to the **HIBDATA** register at offsets 0x030-0x12C.
4. Set the RTC Match Wake-Up and start the hibernation sequence by writing 0x0000.004F to the **HIBCTL** register at offset 0x010.

7.4.4 External Wake-Up from Hibernation

Use the following steps to implement the Hibernation module with the external $\overline{\text{WAKE}}$ pin as the wake-up source for the microcontroller:

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

Note that in this mode, if the RTC is disabled, then the Hibernation clock source is powered down during Hibernation mode and is powered up again on the external wake event to save power during hibernation. If the RTC is enabled before hibernation, it will continue to operate during hibernation.

7.4.5 RTC or External Wake-Up from Hibernation

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

7.4.6 Register Reset

The Hibernation module handles resets according to the following conditions:

- Cold Reset

  When the hibernation module has no externally applied voltage and detects a change to either V_{DD} or V_{BAT}, it resets all hibernation module registers to the value in Table 7-3 on page 177.

- Reset During Hibernation Module Disable

  When the module has either not been enabled or has been disabled by software, the reset is passed through to the Hibernation module circuitry, and the internal state of the module is reset. Non-volatile memory contents are not reset to zero and contents after reset are indeterminate.

- Reset While Hibernation Module is in Hibernation Mode

  While in Hibernation mode, or while transitioning from Hibernation mode to run mode (leaving the power cut), the reset generated by the POR circuitry of the microcontroller is suppressed, and the state of the Hibernation module's registers is unaffected.

- Reset While Hibernation Module is in Normal Mode

While in normal mode (not hibernating), any reset is suppressed if either the `RTCEN` or the `PINWEN` bit is set in the **HIBCTL** register, and the content/state of the control and data registers is unaffected.

Software must initialize any control or data registers in this condition. Therefore, software is the only mechanism to enable or disable the oscillator and real-time clock operation, or to clear contents of the data memory. The only state that must be cleared by a reset operation while not in Hibernation mode is any state that prevents software from managing the interface.

7.5 Register Map

Table 7-3 on page 177 lists the Hibernation registers. All addresses given are relative to the Hibernation Module base address at 0x400F.C000. Note that the Hibernation module clock must be enabled before the registers can be programmed (see page 141).

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the `WRC` bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 170.

Important: Reset values apply only to a cold reset. Once configured, the Hibernate module ignores any system reset as long as V_{BAT} is present.

Table 7-3. Hibernation Module Register Map

Offset	Name	Type	Reset	Description	See page
0x000	HIBRTCC	RO	0x0000.0000	Hibernation RTC Counter	178
0x004	HIBRTCM0	R/W	0xFFFF.FFFF	Hibernation RTC Match 0	179
0x008	HIBRTCM1	R/W	0xFFFF.FFFF	Hibernation RTC Match 1	180
0x00C	HIBRTCLD	R/W	0xFFFF.FFFF	Hibernation RTC Load	181
0x010	HIBCTL	R/W	0x8000.0000	Hibernation Control	182
0x014	HIBIM	R/W	0x0000.0000	Hibernation Interrupt Mask	185
0x018	HIBRIS	RO	0x0000.0000	Hibernation Raw Interrupt Status	187
0x01C	HIBMIS	RO	0x0000.0000	Hibernation Masked Interrupt Status	189
0x020	HIBIC	R/W1C	0x0000.0000	Hibernation Interrupt Clear	191
0x024	HIBRTCT	R/W	0x0000.7FFF	Hibernation RTC Trim	192
0x030-0x12C	HIBDATA	R/W	-	Hibernation Data	193

7.6 Register Descriptions

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

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

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

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the `WRC` bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 170.

Hibernation RTC Counter (HIBRTCC)

Base 0x400F.C000
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.

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the `WRC` bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See "Register Access Timing" on page 170.

Hibernation RTC Match 0 (HIBRTCM0)

Base 0x400F.C000
Offset 0x004
Type R/W, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RTCM0															
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
	RTCM0															
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	RTCM0	R/W	0xFFFF.FFFF	RTC Match 0 A write loads the value into the RTC match register. A read returns the current match value.

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

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

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the `WRC` bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See "Register Access Timing" on page 170.

Hibernation RTC Match 1 (HIBRTCM1)

Base 0x400F.C000
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 used to load a 32-bit value loaded into the RTC counter. The load occurs immediately upon this register being written.

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the `WRC` bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 170.

Hibernation RTC Load (HIBRTCLD)

Base 0x400F.C000
Offset 0x00C
Type R/W, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	RTCLD															
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
	RTCLD															
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	RTCLD	R/W	0xFFFF.FFFF	RTC Load A write loads the current value into the RTC counter (`RTCC`). A read returns the 32-bit load value.

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

This register is the control register for the Hibernation module.

Hibernation Control (HIBCTL)

Base 0x400F.C000
Offset 0x010
Type R/W, reset 0x8000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WRC	reserved														
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved							VDD3ON	VABORT	CLK32EN	LOWBATEN	PINWEN	RTCWEN	CLKSEL	HIBREQ	RTCEN
Type	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	WRC	RO	1	Write Complete/Capable Value Description 0 The interface is processing a prior write and is busy. Any write operation that is attempted while WRC is 0 results in undetermined behavior. 1 The interface is ready to accept a write. Software must poll this bit between write requests and defer writes until `WRC`=1 to ensure proper operation. This difference may be exploited by software at reset time to detect which method of programming is appropriate: 0 = software delay loops required; 1 = `WRC` paced available. The bit name `WRC` means "Write Complete," which is the normal use of the bit (between write accesses). However, because the bit is set out-of-reset, the name can also mean "Write Capable" which simply indicates that the interface may be written to by software.
30:9	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	VDD3ON	R/W	0	VDD Powered Value Description 1 The internal switches control the power to the on-chip modules (VDD3ON mode). 0 The internal switches are not used. The `HIB` signal should be used to control an external switch or regulator. Note that regardless of the status of the `VDD3ON` bit, the `HIB` signal is asserted during Hibernate mode. Thus, when `VDD3ON` is set, the `HIB` signal should not be connected to the 3.3V regulator, and the 3.3V power source should remain connected.

Bit/Field	Name	Type	Reset	Description
7	VABORT	R/W	0	Power Cut Abort Enable Value – Description 1 – Power cut is aborted. 0 – A power cut occurs during a low-battery alert.
6	CLK32EN	R/W	0	Clocking Enable This bit must be enabled to use the Hibernation module. Value – Description 1 – The clock source to the Hibernation module is enabled. 0 – The clock source to the Hibernation module is disabled. The `CLKSEL` bit is used to select between the 4.194304-MHz crystal source and the 32.768-kHz oscillator source. If a crystal is used, then software should wait 20 ms after setting this bit to allow the crystal to power up and stabilize.
5	LOWBATEN	R/W	0	Low Battery Monitoring Enable Value – Description 1 – Low battery voltage detection is enabled. If $V_{BAT} < V_{LOWBAT}$, the `LOWBAT` bit in the **HIBRIS** register is set. 0 – Low battery monitoring is disabled.
4	PINWEN	R/W	0	External $\overline{\text{WAKE}}$ Pin Enable Value – Description 1 – An assertion of the $\overline{\text{WAKE}}$ pin takes the microcontroller out of hibernation. 0 – The status of the $\overline{\text{WAKE}}$ pin has no effect on hibernation.
3	RTCWEN	R/W	0	RTC Wake-up Enable Value – Description 1 – An RTC match event (the value the **HIBRTCC** register matches the value of the **HIBRTCM0** or **HIBRTCM1** register) takes the microcontroller out of hibernation. 0 – An RTC match event has no effect on hibernation.
2	CLKSEL	R/W	0	Hibernation Module Clock Select Value – Description 1 – Use raw output. Use this value for a 32.768-kHz oscillator. 0 – Use Divide-by-128 output. Use this value for a 4.194304-MHz crystal.

Bit/Field	Name	Type	Reset	Description
1	HIBREQ	R/W	0	Hibernation Request Value Description 1 Set this bit to initiate hibernation. 0 No hibernation request. After a wake-up event, this bit is automatically cleared by hardware.
0	RTCEN	R/W	0	RTC Timer Enable Value Description 1 The Hibernation module RTC is enabled. The RTC remains active during hibernation. 0 The Hibernation module RTC is disabled. If PINWEN is set, enabling an external wake event, the RTC stops during hibernation to save power.

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

This register is the interrupt mask register for the Hibernation module interrupt sources. Each bit in this register masks the corresponding bit in the **Hibernation Raw Interrupt Status (HIBRIS)** register. If a bit is unmasked, the interrupt is sent to the interrupt controller. If the bit is masked, the interrupt is not sent to the interrupt controller.

Hibernation Interrupt Mask (HIBIM)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	R/W	0	External Wake-Up Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `EXTW` bit in the **HIBRIS** register is set. 0 The `EXTW` interrupt is suppressed and not sent to the interrupt controller.
2	LOWBAT	R/W	0	Low Battery Voltage Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `LOWBAT` bit in the **HIBRIS** register is set. 0 The `LOWBAT` interrupt is suppressed and not sent to the interrupt controller.
1	RTCALT1	R/W	0	RTC Alert 1 Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `RTCALT1` bit in the **HIBRIS** register is set. 0 The `RTCALT1` interrupt is suppressed and not sent to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
0	RTCALT0	R/W	0	RTC Alert 0 Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `RTCALT0` bit in the **HIBRIS** register is set. 0 The `RTCALT0` interrupt is suppressed and not sent to the interrupt controller.

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

This register is the raw interrupt status for the Hibernation module interrupt sources. Each bit can be masked by clearing the corresponding bit in the **HIBIM** register. When a bit is masked, the interrupt is not sent to the interrupt controller. Bits in this register are cleared by writing a 1 to the corresponding bit in the **Hibernation Interrupt Clear (HIBIC)** register.

Hibernation Raw Interrupt Status (HIBRIS)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	RO	0	External Wake-Up Raw Interrupt Status Value Description 1 The $\overline{\texttt{WAKE}}$ pin has been asserted. 0 The $\overline{\texttt{WAKE}}$ pin has not been asserted. This bit is cleared by writing a 1 to the `EXTW` bit in the **HIBIC** register.
2	LOWBAT	RO	0	Low Battery Voltage Raw Interrupt Status Value Description 1 The battery voltage dropped below V_{LOWBAT}. 0 The battery voltage has not dropped below V_{LOWBAT}. This bit is cleared by writing a 1 to the `LOWBAT` bit in the **HIBIC** register.
1	RTCALT1	RO	0	RTC Alert 1 Raw Interrupt Status Value Description 1 The value of the **HIBRTCC** register matches the value in the **HIBRTCM1** register. 0 No match This bit is cleared by writing a 1 to the `RTCALT1` bit in the **HIBIC** register.

Bit/Field	Name	Type	Reset	Description
0	RTCALT0	RO	0	RTC Alert 0 Raw Interrupt Status Value Description 1 The value of the **HIBRTCC** register matches the value in the **HIBRTCM0** register. 0 No match This bit is cleared by writing a 1 to the `RTCALT0` bit in the **HIBIC** register.

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

This register is the masked interrupt status for the Hibernation module interrupt sources. Bits in this register are the AND of the corresponding bits in the **HIBRIS** and **HIBIM** registers. When both corresponding bits are set, the bit in this register is set, and the interrupt is sent to the interrupt controller.

Hibernation Masked Interrupt Status (HIBMIS)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												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	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	RO	0	External Wake-Up Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a $\overline{\text{WAKE}}$ pin assertion. 0 An external wake-up interrupt has not occurred. This bit is cleared by writing a 1 to the `EXTW` bit in the **HIBIC** register.
2	LOWBAT	RO	0	Low Battery Voltage Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a low battery voltage condition. 0 A low battery voltage interrupt has not occurred. This bit is cleared by writing a 1 to the `LOWBAT` bit in the **HIBIC** register.
1	RTCALT1	RO	0	RTC Alert 1 Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a low battery voltage condition. 0 A low battery voltage interrupt has not occurred. When this bit is set, an RTC match 1 interrupt is sent to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
0	RTCALT0	RO	0	RTC Alert 0 Masked Interrupt Status When this bit is set, an RTC match 0 interrupt is sent to the interrupt controller.

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

This register is the interrupt write-one-to-clear register for the Hibernation module interrupt sources. Writing a 1 to a bit clears the corresponding interrupt in the **HIBRIS** register.

Hibernation Interrupt Clear (HIBIC)

Base 0x400F.C000
Offset 0x020
Type R/W1C, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	R/W1C	0	External Wake-Up Masked Interrupt Clear Writing a 1 to this bit clears the `EXTW` bit in the **HIBRIS** and **HIBMIS** registers. Reads return an indeterminate value.
2	LOWBAT	R/W1C	0	Low Battery Voltage Masked Interrupt Clear Writing a 1 to this bit clears the `LOWBAT` bit in the **HIBRIS** and **HIBMIS** registers. Reads return an indeterminate value.
1	RTCALT1	R/W1C	0	RTC Alert1 Masked Interrupt Clear Writing a 1 to this bit clears the `RTCALT1` bit in the **HIBRIS** and **HIBMIS** registers. Reads return an indeterminate value.
0	RTCALT0	R/W1C	0	RTC Alert0 Masked Interrupt Clear Writing a 1 to this bit clears the `RTCALT0` bit in the **HIBRIS** and **HIBMIS** registers. Reads return an indeterminate value.

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

This register contains the value that is used to trim the RTC clock predivider. It represents the computed underflow value that is used during the trim cycle. It is represented as 0x7FFF ± N clock cycles, where N is the number of clock cycles to add or subtract every 63 seconds.

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the `WRC` bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See "Register Access Timing" on page 170.

Hibernation RTC Trim (HIBRTCT)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	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. Compensation can be adjusted by software by moving the default value of 0x7FFF up or down. Moving the value up slows down the RTC and moving the value down speeds up the RTC.

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

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

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and have special timing requirements. Software should make use of the `WRC` bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See "Register Access Timing" on page 170.

Hibernation Data (HIBDATA)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RTD															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:0	RTD	R/W	-	Hibernation Module NV Data

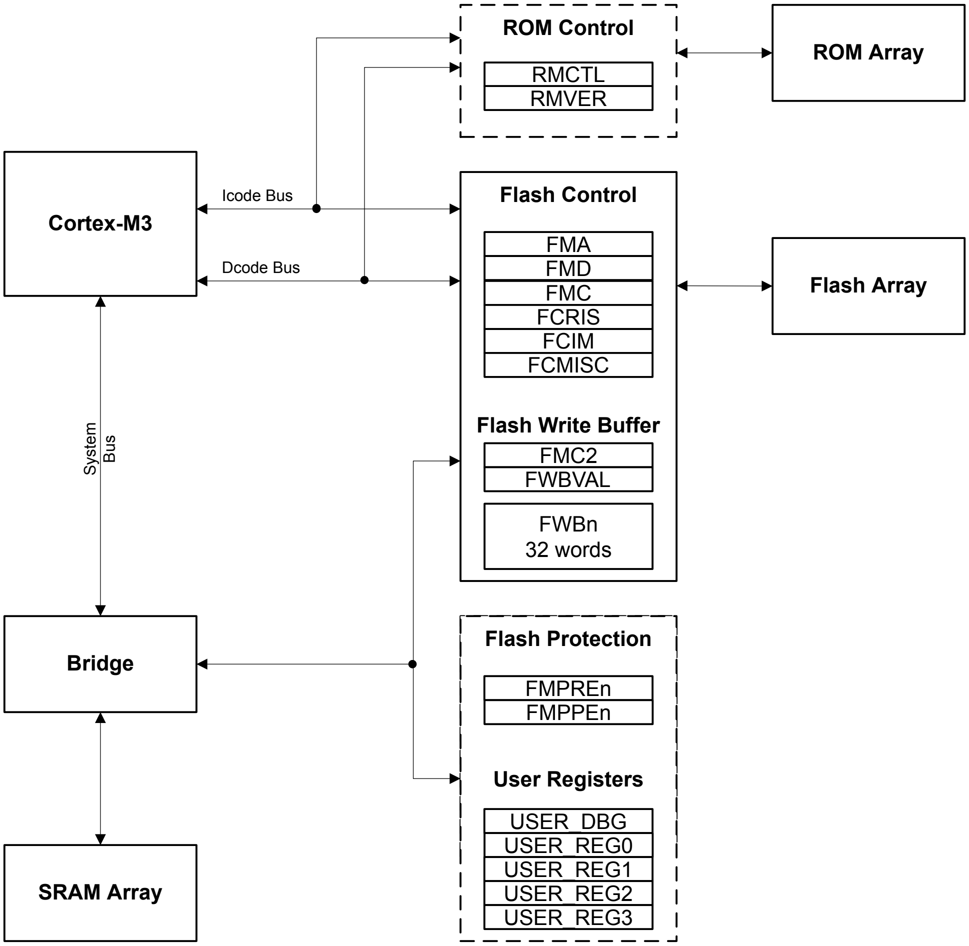
8 Internal Memory

The LM3S1W16 microcontroller comes with 8 KB of bit-banded SRAM, internal ROM, and 32 KB of Flash memory. The Flash memory controller provides a user-friendly interface, making Flash memory programming a simple task. Flash memory protection can be applied to the Flash memory on a 2-KB block basis.

8.1 Block Diagram

Figure 8-1 on page 194 illustrates the internal memory blocks and control logic. The dashed boxes in the figure indicate registers residing in the System Control module.

Figure 8-1. Internal Memory Block Diagram



8.2 Functional Description

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

Note: The μDMA controller can transfer data to and from the on-chip SRAM. However, because the Flash memory and ROM are located on a separate internal bus, it is not possible to transfer data from the Flash memory or ROM with the μDMA controller.

8.2.1 SRAM

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

The internal SRAM of the Stellaris® devices is located at address 0x2000.0000 of the device memory map. To reduce the number of time consuming read-modify-write (RMW) operations, ARM has introduced *bit-banding* technology in the Cortex-M3 processor. With a bit-band-enabled processor, certain regions in the memory map (SRAM and peripheral space) can use address aliases to access individual bits in a single, atomic operation. The bit-band base is located at address 0x2200.0000.

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

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

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

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

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

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

8.2.2 ROM

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

- Stellaris® Boot Loader and vector table (see “Boot Loader” on page 658)
- Stellaris® Peripheral Driver Library (DriverLib) release for product-specific peripherals and interfaces (see “ROM DriverLib Functions” on page 663)

8.2.3 Flash Memory

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

Caution – In systems where power might be cycled more frequently than every five minutes, power must be removed from the microcontroller in a controlled manner to ensure proper operation. Software must request permission to power down the part using the USDREQ bit in the Flash Control (FCTL) register and wait to receive an acknowledge from the USDACK bit prior to removing power.

Note that this power-down process is not required if the microcontroller enters hibernation mode prior to power being removed.

8.2.3.1 Flash Memory Protection

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

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

The policies may be combined as shown in Table 8-1 on page 196.

Table 8-1. Flash Memory Protection Policy Combinations

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

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

The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. These settings create a policy of open access and programmability. The register bits may be changed by clearing the specific register bit. The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The changes are committed using the **Flash Memory Control (FMC)** register. Details on programming these bits are discussed in “Nonvolatile Register Programming” on page 198.

8.3 Flash Memory Initialization and Configuration

8.3.1 Flash Memory Programming

The Stellaris® devices provide a user-friendly interface for Flash memory programming. All erase/program operations are handled via three registers: **Flash Memory Address (FMA)**, **Flash Memory Data (FMD)**, and **Flash Memory Control (FMC)**. Note that if the debug capabilities of the microcontroller have been deactivated, resulting in a "locked" state, a recovery sequence must be performed in order to reactivate the debug module. See “Recovering a "Locked" Microcontroller” on page 71.

Caution – The Flash memory is divided into sectors of electrically separated address ranges of 4 KB each, aligned on 4 KB boundaries. Erase/program operations on a 1-KB page have an electrical effect on the other three 1-KB pages within the sector. A specific 1-KB page must be erased after 6 total erase/program cycles occur to the other pages within it's 4-KB sector. The following sequence of operations on a 4-KB sector of Flash memory (Page 0..3) provides an example:

- **Page 3 is erase and programmed with values.**
- **Page 0, Page 1, and Page 2 are erased and then programmed with values. At this point Page 3 has been affected by 3 erase/program cycles.**
- **Page 0, Page 1, and Page 2 are again erased and then programmed with values. At this point Page 3 has been affected by 6 erase/program cycles.**
- **If the contents of Page 3 must continue to be valid, Page 3 must be erased and reprogrammed before any other page in this sector has another erase or program operation.**

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 memory write key and the `WRITE` bit (a value of 0xA442.0001) to the **FMC** register.
4. Poll the **FMC** register until the `WRITE` bit is cleared.

Important: To ensure proper operation, two writes to the same word must be separated by an ERASE. The following two sequences are allowed:

- ERASE -> PROGRAM value -> PROGRAM 0x0000.0000
- ERASE -> PROGRAM value -> ERASE

The following sequence is NOT allowed:

- ERASE -> PROGRAM value -> PROGRAM value

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 memory write key and the `ERASE` bit (a value of 0xA442.0002) to the **FMC** register.
3. Poll the **FMC** register until the `ERASE` bit is cleared.

8.3.1.3 To perform a mass erase of the Flash memory

1. Write the Flash memory write key and the `MERASE` bit (a value of 0xA442.0004) to the **FMC** register.

2. Poll the **FMC** register until the `MERASE` bit is cleared.

8.3.2 32-Word Flash Memory Write Buffer

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

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

8.3.2.1 To program 32 words with a single buffered Flash memory write operation

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

8.3.3 Nonvolatile Register Programming

This section discusses how to update registers that are resident within the Flash memory itself. These registers exist in a separate space from the main Flash memory array and are not affected by an ERASE or MASS ERASE operation. The bits in these registers can be changed from 1 to 0 with a write operation. The register contents are unaffected by any reset condition except power-on reset, which returns the register contents to 0xFFFF.FFFF. By committing the register values using the `COMT` bit in the **FMC** register, the register contents become nonvolatile and are therefore retained following power cycling. Once the register contents are committed, the only way to restore the factory default values is to perform the sequence described in “Recovering a "Locked" Microcontroller” on page 71.

With the exception of the **USER_DBG** register, the settings in these registers can be tested before committing them to Flash memory. For the **USER_DBG** register, the data to be written is loaded into the **FMD** register before it is committed. The **FMD** register is read only and does not allow the **USER_DBG** operation to be tried before committing it to nonvolatile memory.

Important: The Flash memory resident registers can only have bits changed from 1 to 0 by user programming and can only be committed once. After being committed, these registers can only be restored to their factory default values only by performing the sequence described in “Recovering a "Locked" Microcontroller” on page 71. The mass erase of the main Flash memory array caused by the sequence is performed prior to restoring these registers.

In addition, the **USER_REG0**, **USER_REG1**, **USER_REG2**, **USER_REG3**, and **USER_DBG** registers each use bit 31 (`NW`) to indicate that they have not been committed and bits in the register may be changed from 1 to 0. Table 8-2 on page 199 provides the **FMA** address required for commitment of each of the registers and the source of the data to be written when the **FMC** register is written with a value of 0xA442.0008. After writing the `COMT` bit, the user may poll the **FMC** register to wait for the commit operation to complete.

Table 8-2. User-Programmable Flash Memory Resident Registers

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

8.4 Register Map

Table 8-3 on page 199 lists the ROM Controller register and the Flash memory and control registers. The offset listed is a hexadecimal increment to the register's address. The **FMA**, **FMD**, **FMC**, **FCRIS**, **FCIM**, **FCMISC**, **FMC2**, **FWBVAL**, and **FWBn** register offsets are relative to the Flash memory control base address of 0x400F.D000. The ROM and Flash memory protection register offsets are relative to the System Control base address of 0x400F.E000.

Table 8-3. Flash Register Map

Offset	Name	Type	Reset	Description	See page
Flash Memory Registers (Flash Control Offset)					
0x000	FMA	R/W	0x0000.0000	Flash Memory Address	201
0x004	FMD	R/W	0x0000.0000	Flash Memory Data	202
0x008	FMC	R/W	0x0000.0000	Flash Memory Control	203
0x00C	FCRIS	RO	0x0000.0000	Flash Controller Raw Interrupt Status	205
0x010	FCIM	R/W	0x0000.0000	Flash Controller Interrupt Mask	206
0x014	FCMISC	R/W1C	0x0000.0000	Flash Controller Masked Interrupt Status and Clear	207
0x020	FMC2	R/W	0x0000.0000	Flash Memory Control 2	208
0x030	FWBVAL	R/W	0x0000.0000	Flash Write Buffer Valid	209
0x0F8	FCTL	R/W	0x0000.0000	Flash Control	211
0x100 - 0x17C	FWBn	R/W	0x0000.0000	Flash Write Buffer n	210
Memory Registers (System Control Offset)					
0x0F0	RMCTL	R/W1C	-	ROM Control	212
0x0F4	RMVER	RO	0x0505.0400	ROM Version Register	213
0x130	FMPRE0	R/W	0x0000.FFFF	Flash Memory Protection Read Enable 0	214

Table 8-3. Flash Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x200	FMPRE0	R/W	0x0000.FFFF	Flash Memory Protection Read Enable 0	214
0x134	FMPPE0	R/W	0x0000.FFFF	Flash Memory Protection Program Enable 0	215
0x400	FMPPE0	R/W	0x0000.FFFF	Flash Memory Protection Program Enable 0	215
0x1D0	USER_DBG	R/W	0xFFFF.FFFE	User Debug	216
0x1E0	USER_REG0	R/W	0xFFFF.FFFF	User Register 0	217
0x1E4	USER_REG1	R/W	0xFFFF.FFFF	User Register 1	218
0x1E8	USER_REG2	R/W	0xFFFF.FFFF	User Register 2	219
0x1EC	USER_REG3	R/W	0xFFFF.FFFF	User Register 3	220
0x204	FMPRE1	R/W	0x0000.0000	Flash Memory Protection Read Enable 1	221
0x208	FMPRE2	R/W	0x0000.0000	Flash Memory Protection Read Enable 2	222
0x20C	FMPRE3	R/W	0x0000.0000	Flash Memory Protection Read Enable 3	223
0x404	FMPPE1	R/W	0x0000.0000	Flash Memory Protection Program Enable 1	224
0x408	FMPPE2	R/W	0x0000.0000	Flash Memory Protection Program Enable 2	225
0x40C	FMPPE3	R/W	0x0000.0000	Flash Memory Protection Program Enable 3	226

8.5 Flash Memory Register Descriptions (Flash Control Offset)

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

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

During a write operation, this register contains a 4-byte-aligned address and specifies where the data is written. During erase operations, this register contains a 1 KB-aligned 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
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	OFFSET														
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	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14:0	OFFSET	R/W	0x0	Address Offset Address offset in Flash memory where operation is performed, except for nonvolatile registers (see "Nonvolatile Register Programming" on page 198 for details on values for this field).

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

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

Flash Memory Data (FMD)

Base 0x400F.D000
Offset 0x004
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

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

When this register is written, the Flash memory controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 201). If the access is a write access, the data contained in the **Flash Memory Data (FMD)** register (see page 202) is written to the specified address.

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

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

Flash Memory Control (FMC)

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

	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	0x0000	Flash Memory Write Key This field contains a write key, which is used to minimize the incidence of accidental Flash memory writes. The value 0xA442 must be written into this field for a Flash memory write to occur. Writes to the **FMC** register without this `WRKEY` value are ignored. A read of this field returns the value 0.
15:4	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	COMT	R/W	0	Commit Register Value This bit is used to commit writes to Flash-memory-resident registers and to monitor the progress of that process. Value Description 1 Set this bit to commit (write) the register value to a Flash-memory-resident register. When read, a 1 indicates that the previous commit access is not complete. 0 A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous commit access is complete. A commit can take up to 50 µs. See "Nonvolatile Register Programming" on page 198 for more information on programming Flash-memory-resident registers.

Bit/Field	Name	Type	Reset	Description
2	MERASE	R/W	0	Mass Erase Flash Memory This bit is used to mass erase the Flash main memory and to monitor the progress of that process. Value Description 1 Set this bit to erase the Flash main memory. When read, a 1 indicates that the previous mass erase access is not complete. 0 A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous mass erase access is complete. A mass erase can take up to 250 ms.
1	ERASE	R/W	0	Erase a Page of Flash Memory This bit is used to erase a page of Flash memory and to monitor the progress of that process. Value Description 1 Set this bit to erase the Flash memory page specified by the contents of the **FMA** register. When read, a 1 indicates that the previous page erase access is not complete. 0 A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous page erase access is complete. A page erase can take up to 25 ms.
0	WRITE	R/W	0	Write a Word into Flash Memory This bit is used to write a word into Flash memory and to monitor the progress of that process. Value Description 1 Set this bit to write the data stored in the **FMD** register into the Flash memory location specified by the contents of the **FMA** register. When read, a 1 indicates that the write update access is not complete. 0 A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous write update access is complete. Writing a single word can take up to 50 µs.

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

This register indicates that the Flash memory controller has an interrupt condition. An interrupt is sent to the interrupt controller only if the corresponding **FCIM** register bit is set.

Flash Controller Raw Interrupt Status (FCRIS)

Base 0x400F.D000
Offset 0x00C
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														PRIS	ARIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PRIS	RO	0	Programming Raw Interrupt Status This bit provides status on programming cycles which are write or erase actions generated through the **FMC** or **FMC2** register bits (see page 203 and page 208). Value Description 1 The programming cycle has completed. 0 The programming cycle has not completed. This status is sent to the interrupt controller when the `PMASK` bit in the **FCIM** register is set. This bit is cleared by writing a 1 to the `PMISC` bit in the **FCMISC** register.
0	ARIS	RO	0	Access Raw Interrupt Status This bit indicates if the Flash memory was improperly accessed. Value Description 1 The program tried to access the Flash memory counter to the policy set in the **FMPREn** and **FMPPEn** registers. 0 No access has tried to improperly access the Flash memory. This status is sent to the interrupt controller when the `AMASK` bit in the **FCIM** register is set. This bit is cleared by writing a 1 to the `AMISC` bit in the **FCMISC** register.

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

This register controls whether the Flash memory controller generates interrupts to the controller.

Flash Controller Interrupt Mask (FCIM)

Base 0x400F.D000
Offset 0x010
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PMASK	R/W	0	Programming Interrupt Mask This bit controls the reporting of the programming raw interrupt status to the interrupt controller. Value Description 1 An interrupt is sent to the interrupt controller when the `PRIS` bit is set. 0 The `PRIS` interrupt is suppressed and not sent to the interrupt controller.
0	AMASK	R/W	0	Access Interrupt Mask This bit controls the reporting of the access raw interrupt status to the interrupt controller. Value Description 1 An interrupt is sent to the interrupt controller when the `ARIS` bit is set. 0 The `ARIS` interrupt is suppressed and not sent to the interrupt controller.

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

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

Flash Controller Masked Interrupt Status and Clear (FCMISC)

Base 0x400F.D000
Offset 0x014
Type R/W1C, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														PMISC	AMISC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PMISC	R/W1C	0	Programming Masked Interrupt Status and Clear Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because a programming cycle completed. Writing a 1 to this bit clears `PMISC` and also the `PRIS` bit in the **FCRIS** register (see page 205). 0 When read, a 0 indicates that a programming cycle complete interrupt has not occurred. A write of 0 has no effect on the state of this bit.
0	AMISC	R/W1C	0	Access Masked Interrupt Status and Clear Value Description 1 When read, a 1 indicates that an unmasked interrupt was signaled because an improper access to protected Flash memory was attempted. Writing a 1 to this bit clears `AMISC` and also the `ARIS` bit in the **FCRIS** register (see page 205). 0 When read, a 0 indicates that no improper accesses have occurred. A write of 0 has no effect on the state of this bit.

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

When this register is written, the Flash memory controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 201). If the access is a write access, the data contained in the **Flash Write Buffer (FWB)** registers is written.

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

Flash Memory Control 2 (FMC2)

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

	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															WRBUF
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
31:16	WRKEY	WO	0x0000	Flash Memory Write Key This field contains a write key, which is used to minimize the incidence of accidental Flash memory writes. The value 0xA442 must be written into this field for a write to occur. Writes to the **FMC2** register without this `WRKEY` value are ignored. A read of this field returns the value 0.
15:1	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WRBUF	R/W	0	Buffered Flash Memory Write This bit is used to start a buffered write to Flash memory. Value Description 1 Set this bit to write the data stored in the **FWBn** registers to the location specified by the contents of the **FMA** register. When read, a 1 indicates that the previous buffered Flash memory write access is not complete. 0 A write of 0 has no effect on the state of this bit. When read, a 0 indicates that the previous buffered Flash memory write access is complete. A buffered Flash memory write can take up to 4 ms.

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

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

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

Flash Write Buffer Valid (FWBVAL)

Base 0x400F.D000
Offset 0x030
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	FWB[n]															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	FWB[n]	R/W	0x0	Flash Memory Write Buffer

Value	Description
1	The corresponding **FWBn** register has been updated since the last buffer write operation and is ready to be written to Flash memory.
0	The corresponding **FWBn** register has no new data to be written.

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

Register 9: Flash Write Buffer n (FWBn), offset 0x100 - 0x17C

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

Flash Write Buffer n (FWBn)

Base 0x400F.D000
Offset 0x100 - 0x17C
Type R/W, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:0	DATA	R/W	0x0000.0000	Data Data to be written into the Flash memory.

Register 10: Flash Control (FCTL), offset 0x0F8

This register is used to ensure that the microcontroller is powered down in a controlled fashion in systems where power is cycled more frequently than once every five minutes. The `USDREQ` bit should be set to indicate that power is going to be turned off. Software should poll the `USDACK` bit to determine when it is acceptable to power down.

Note that this power-down process is not required if the microcontroller enters hibernation mode prior to power being removed.

Flash Control (FCTL)

Base 0x400F.D000
Offset 0x0F8
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														USDACK	USDREQ
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
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	USDACK	RO	0	User Shut Down Acknowledge Value Description 1 The microcontroller can be powered down. 0 The microcontroller cannot yet be powered down. This bit should be set within 50 ms of setting the `USDREQ` bit.
0	USDREQ	R/W	0	User Shut Down Request Value Description 1 Requests permission to power down the microcontroller. 0 No effect.

8.6 Memory Register Descriptions (System Control Offset)

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

Register 11: ROM Control (RMCTL), offset 0x0F0

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

ROM Control (RMCTL)

Base 0x400F.E000
Offset 0x0F0
Type R/W1C, reset -

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															BA
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	BA	R/W1C	-	Boot Alias Upon reset, the system control module checks address 0x000.0004 to see if it has a valid reset vector. If the data at address 0x0000.0004 is 0xFFFF.FFFF, then it is assumed that the Flash memory has not yet been programmed, and this bit is then set by hardware so that the on-chip ROM appears at address 0x0. Value Description 1 The microcontroller's ROM appears at address 0x0. This bit is set automatically if the data at address 0x0000.0004 is 0xFFFF.FFFF. 0 The Flash memory is at address 0x0. This bit is cleared by writing a 1 to this bit position.

Register 12: ROM Version Register (RMVER), offset 0x0F4

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

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

ROM Version Register (RMVER)

Base 0x400F.E000
Offset 0x0F4
Type RO, reset 0x0505.0400

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CONT								SIZE							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	VER								REV							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:24	CONT	RO	0x05	ROM Contents Value Description 0x05 Stellaris Boot Loader & DriverLib with AES
23:16	SIZE	RO	0x00 0x05	ROM Size This field encodes the size of the ROM. Value Description 0x00 Stellaris Boot Loader & DriverLib 0x05 Stellaris Boot Loader & DriverLib with AES
15:8	VER	RO	0x104	ROM Version
7:0	REV	RO	0x0	ROM Revision

Register 13: Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200

Note: This register is aliased for backwards compatability.

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

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 0 (FMPRE0)

Base 0x400F.E000
Offset 0x130 and 0x200
Type R/W, reset 0x0000.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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	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	0x0000FFFF	Flash Read Enable Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.

Value	Description
0x0000FFFF	Bits [15:0] each enable protection on a 2-KB block of Flash memory up to the total of 32 KB.

Register 14: Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400

Note: This register is aliased for backwards compatability.

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

This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 0 (FMPPE0)

Base 0x400F.E000
Offset 0x134 and 0x400
Type R/W, reset 0x0000.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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PROG_ENABLE															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	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	0x0000FFFF	Flash Programming Enable Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.

Value	Description
0x0000FFFF	Bits [15:0] each enable protection on a 2-KB block of Flash memory up to the total of 32 KB.

Register 15: 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 NW bit (bit 31) indicates that the register has not yet been committed and is controlled through hardware to ensure that the register is only committed once. Prior to being committed, bits can only be changed from 1 to 0. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. The only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter.

User Debug (USER_DBG)

Base 0x400F.E000
Offset 0x1D0
Type R/W, reset 0xFFFF.FFFE

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	NW	DATA														
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
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														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	NW	R/W	1	User Debug Not Written When set, this bit indicates that this 32-bit register has not been committed. When clear, this bit specifies that this register has been committed and may not be committed again.
30:2	DATA	R/W	0x1FFFFFFF	User Data Contains the user data value. This field is initialized to all 1s and can only be committed once.
1	DBG1	R/W	1	Debug Control 1 The `DBG1` bit must be 1 and `DBG0` must be 0 for debug to be available.
0	DBG0	R/W	0	Debug Control 0 The `DBG1` bit must be 1 and `DBG0` must be 0 for debug to be available.

Register 16: User Register 0 (USER_REG0), offset 0x1E0

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

This register provides 31 bits of user-defined data that is non-volatile and can only be committed once. Bit 31 indicates that the register is available to be committed and is controlled through hardware to ensure that the register is only committed once. Prior to being committed, bits can only be changed from 1 to 0. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. The write-once characteristics of this register are useful for keeping static information like communication addresses that need to be unique per part and would otherwise require an external EEPROM or other non-volatile device. The only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG section.

User Register 0 (USER_REG0)

Base 0x400F.E000
Offset 0x1E0
Type R/W, reset 0xFFFF.FFFF

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

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

Register 17: User Register 1 (USER_REG1), offset 0x1E4

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

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

User Register 1 (USER_REG1)

Base 0x400F.E000
Offset 0x1E4
Type R/W, reset 0xFFFF.FFFF

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

Register 18: User Register 2 (USER_REG2), offset 0x1E8

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

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

User Register 2 (USER_REG2)

Base 0x400F.E000
Offset 0x1E8
Type R/W, reset 0xFFFF.FFFF

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

Register 19: User Register 3 (USER_REG3), offset 0x1EC

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

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

User Register 3 (USER_REG3)

Base 0x400F.E000
Offset 0x1EC
Type R/W, reset 0xFFFF.FFFF

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

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

Register 20: Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204

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

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 64 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 1 (FMPRE1)

Base 0x400F.E000
Offset 0x204
Type R/W, reset 0x0000.0000

	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0x00000000	Flash Read Enable Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.

Value	Description
0x00000000	Bits [15:0] each enable protection on a 2-KB block of Flash memory in the range from 65 to 96 KB.

Register 21: Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208

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

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 128 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 2 (FMPRE2)

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

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

Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0x00000000	Flash Read Enable Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.

Value	Description
0x00000000	Bits [15:0] each enable protection on a 2-KB block of Flash memory in the range from 129 to 160 KB.

Register 22: Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C

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

This register stores the read-only protection bits for each 2-KB flash block (**FMPPEn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPREn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Read Enable 3 (FMPRE3)

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

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

Bit/Field	Name	Type	Reset	Description
31:0	READ_ENABLE	R/W	0x00000000	Flash Read Enable Configures 2-KB flash blocks to be read or executed only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”.

Value	Description
0x00000000	Bits [15:0] each enable protection on a 2-KB block of Flash memory in the range from 193 to 224 KB.

Register 23: Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404

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

This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 64 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 1 (FMPPE1)

Base 0x400F.E000
Offset 0x404
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PROG_ENABLE															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Value	Description
0x00000000	Bits [15:0] each enable protection on a 2-KB block of Flash memory in the range from 65 to 96 KB.

Register 24: Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408

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

This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 128 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 2 (FMPPE2)

Base 0x400F.E000
Offset 0x408
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PROG_ENABLE															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Value	Description
0x00000000	Bits [15:0] each enable protection on a 2-KB block of Flash memory in the range from 129 to 160 KB.

Register 25: Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C

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

This register stores the execute-only protection bits for each 2-KB flash block (**FMPREn** stores the execute-only bits). Flash memory up to a total of 64 KB is controlled by this register. Other **FMPPEn** registers (if any) provide protection for other 64K blocks. This register is loaded during the power-on reset sequence. The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This achieves a policy of open access and programmability. The register bits may be changed by writing the specific register bit. However, this register is R/W0; the user can only change the protection bit from a 1 to a 0 (and may NOT change a 0 to a 1). The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. The reset value shown only applies to power-on reset; any other type of reset does not affect this register. Once committed, the only way to restore the factory default value of this register is to perform the "Recover Locked Device" sequence detailed in the JTAG chapter. If the Flash memory size on the device is less than 192 KB, this register usually reads as zeroes, but software should not rely on these bits to be zero. For additional information, see the "Flash Memory Protection" section.

Flash Memory Protection Program Enable 3 (FMPPE3)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PROG_ENABLE															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Value	Description
0x00000000	Bits [15:0] each enable protection on a 2-KB block of Flash memory in the range from 193 to 224 KB.

9 Micro Direct Memory Access (μDMA)

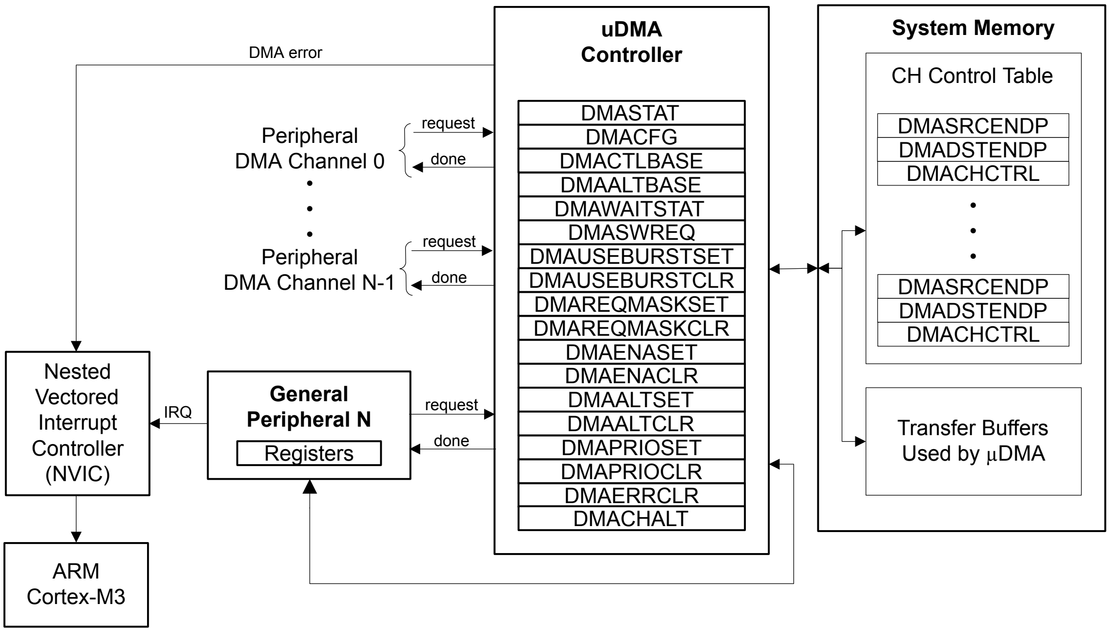
The LM3S1W16 microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (μDMA). The μDMA controller provides a way to offload data transfer tasks from the Cortex-M3 processor, allowing for more efficient use of the processor and the available bus bandwidth. The μDMA controller can perform transfers between memory and peripherals. It has dedicated channels for each supported on-chip module and can be programmed to automatically perform transfers between peripherals and memory as the peripheral is ready to transfer more data. The μDMA controller provides the following features:

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

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

9.1 Block Diagram

Figure 9-1. μDMA Block Diagram



9.2 Functional Description

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

The μDMA controller can transfer data to and from the on-chip SRAM. However, because the Flash memory and ROM are located on a separate internal bus, it is not possible to transfer data from the Flash memory or ROM with the μDMA controller.

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

Each channel also has a configurable arbitration size. The arbitration size is the number of items that are transferred in a burst before the μDMA controller rearbitrates for channel priority. Using the arbitration size, it is possible to control exactly how many items are transferred to or from a peripheral each time it makes a μDMA service request.

9.2.1 Channel Assignments

μDMA channels 0-31 are assigned to peripherals according to the following table. The **DMA Channel Alternate Select (DMACHALT)** register (see page 275) can be used to specify the alternate assignment. Most channels have primary and alternate assignments. If the primary function is not available on this microcontroller, the alternate function becomes the primary function. If the alternate function is not available, the primary function is the only option.

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

Table 9-1. μDMA Channel Assignments

μDMA Channel	Peripheral Assigned	Alternate Assignment
0	Available for software	UART2 Receive
1	Available for software	UART2 Transmit
2	Available for software	Available for software
3	Available for software	Available for software
4	Available for software	General-Purpose Timer 2A
5	Available for software	General-Purpose Timer 2B
6	Available for software	General-Purpose Timer 2A
7	Available for software	General-Purpose Timer 2B
8	UART0 Receive	UART1 Receive
9	UART0 Transmit	UART1 Transmit
10	SSI0 Receive	SSI1 Receive
11	SSI0 Transmit	SSI1 Transmit
12	Available for software	UART2 Receive
13	Available for software	UART2 Transmit
14	ADC0 Sample Sequencer 0	General-Purpose Timer 2A
15	ADC0 Sample Sequencer 1	General-Purpose Timer 2B
16	ADC0 Sample Sequencer 2	Available for software
17	ADC0 Sample Sequencer 3	Available for software
18	General-Purpose Timer 0A	General-Purpose Timer 1A
19	General-Purpose Timer 0B	General-Purpose Timer 1B
20	General-Purpose Timer 1A	Available for software
21	General-Purpose Timer 1B	Available for software
22	UART1 Receive	Available for software
23	UART1 Transmit	Available for software
24	SSI1 Receive	Available for software
25	SSI1 Transmit	Available for software
26	Available for software	Available for software
27	Available for software	Available for software

Table 9-1. µDMA Channel Assignments *(continued)*

µDMA Channel	Peripheral Assigned	Alternate Assignment
28	Available for software	Available for software
29	Available for software	Available for software
30	Dedicated for software use	
31	Reserved	

9.2.2 Priority

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

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

9.2.3 Arbitration Size

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

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

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

9.2.4 Request Types

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

The µDMA controller responds differently depending on whether the peripheral is making a single request or a burst request. If both are asserted, and the µDMA channel has been set up for a burst transfer, then the burst request takes precedence. See Table 9-2, which shows how each peripheral supports the two request types.

Table 9-2. Request Type Support

Peripheral	Single Request Signal	Burst Request Signal
UART TX	TX FIFO Not Full	TX FIFO Level (configurable)

Table 9-2. Request Type Support *(continued)*

Peripheral	Single Request Signal	Burst Request Signal
UART RX	RX FIFO Not Empty	RX FIFO Level (configurable)
SSI TX	TX FIFO Not Full	TX FIFO Level (fixed at 4)
SSI RX	RX FIFO Not Empty	RX FIFO Level (fixed at 4)
ADC	None	Sequencer `IE` bit
General-Purpose Timer	None	Raw interrupt pulse

9.2.4.1 Single Request

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

9.2.4.2 Burst Request

When a burst request is detected, the μDMA controller transfers the number of items that is the lesser of the arbitration size or the number of items remaining in the transfer. Therefore, the arbitration size should be the same as the number of data items that the peripheral can accommodate when making a burst request. For example, the UART generates a burst request based on the FIFO trigger level. In this case, the arbitration size should be set to the amount of data that the FIFO can transfer when the trigger level is reached. A burst transfer runs to completion once it is started, and cannot be interrupted, even by a higher priority channel. Burst transfers complete in a shorter time than the same number of non-burst transfers.

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

9.2.5 Channel Configuration

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

Table 9-3 on page 232 shows the layout in memory of the channel control table. Each channel may have one or two control structures in the control table: a primary control structure and an optional alternate control structure. The table is organized so that all of the primary entries are in the first half of the table, and all the alternate structures are in the second half of the table. The primary entry is used for simple transfer modes where transfers can be reconfigured and restarted after each transfer is complete. In this case, the alternate control structures are not used and therefore only the first half of the table must be allocated in memory; the second half of the control table is not necessary, and that memory can be used for something else. If a more complex transfer mode is used such as ping-pong or scatter-gather, then the alternate control structure is also used and memory space should be allocated for the entire table.

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

Table 9-3. Control Structure Memory Map

Offset	Channel
0x0	0, Primary
0x10	1, Primary
...	...
0x1F0	31, Primary
0x200	0, Alternate
0x210	1, Alternate
...	...
0x3F0	31, Alternate

Table 9-4 shows an individual control structure entry in the control table. Each entry is aligned on a 16-byte boundary. The entry contains four long words: the source end pointer, the destination end pointer, the control word, and an unused entry. The end pointers point to the ending address of the transfer and are inclusive. If the source or destination is non-incrementing (as for a peripheral register), then the pointer should point to the transfer address.

Table 9-4. Channel Control Structure

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

The control word contains the following fields:

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

The control word and each field are described in detail in “μDMA Channel Control Structure” on page 249. The μDMA controller updates the transfer size and transfer mode fields as the transfer is performed. At the end of a transfer, the transfer size indicates 0, and the transfer mode indicates "stopped." Because the control word is modified by the μDMA controller, it must be reconfigured before each new transfer. The source and destination end pointers are not modified, so they can be left unchanged if the source or destination addresses remain the same.

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

9.2.6 Transfer Modes

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

9.2.6.1 Stop Mode

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

9.2.6.2 Basic Mode

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

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

9.2.6.3 Auto Mode

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

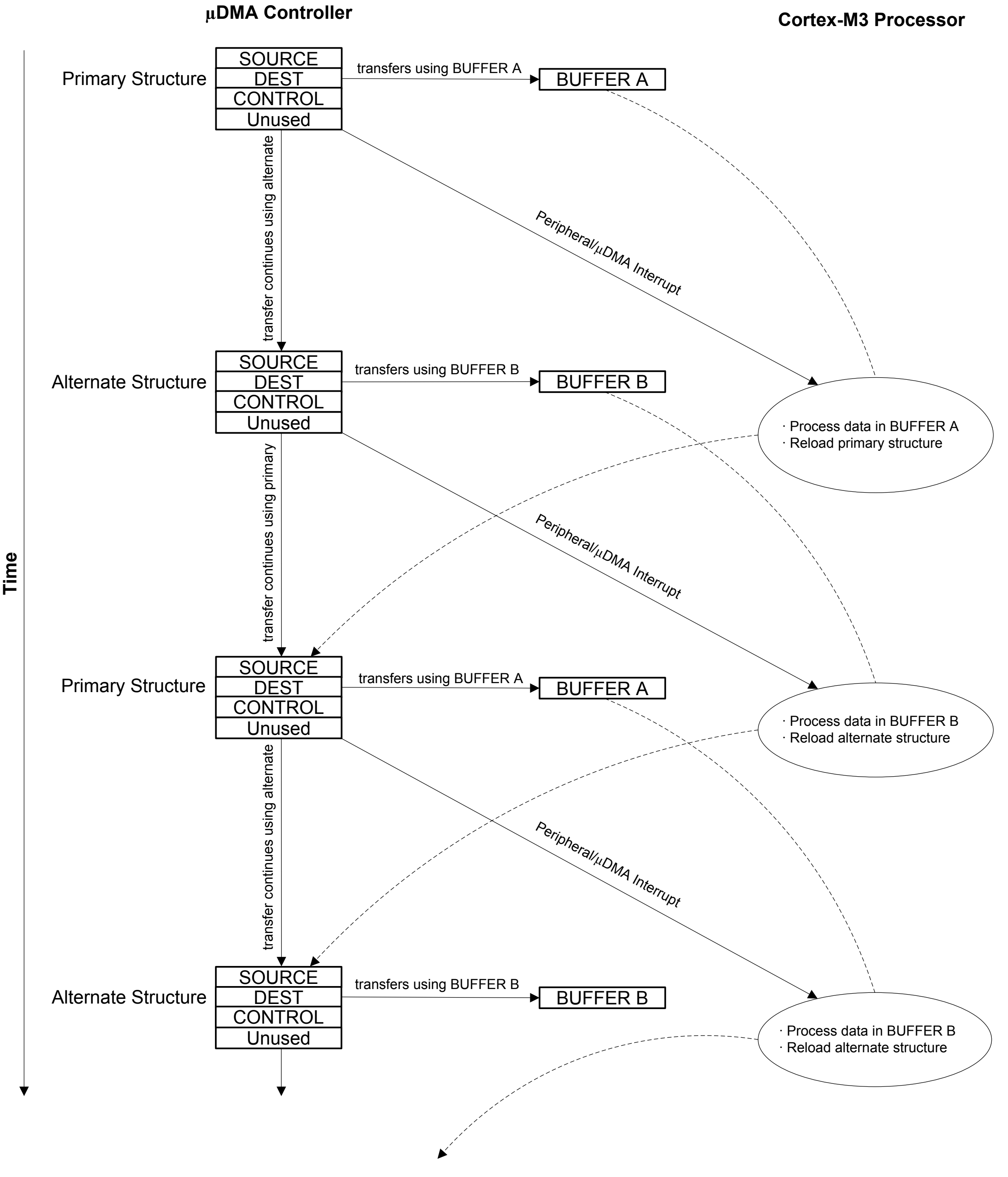
When all the items have been transferred using Auto mode, the µDMA controller sets the mode for that channel to Stop.

9.2.6.4 Ping-Pong

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

Refer to Figure 9-2 for an example showing operation in Ping-Pong mode.

Figure 9-2. Example of Ping-Pong μDMA Transaction



9.2.6.5 Memory Scatter-Gather

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

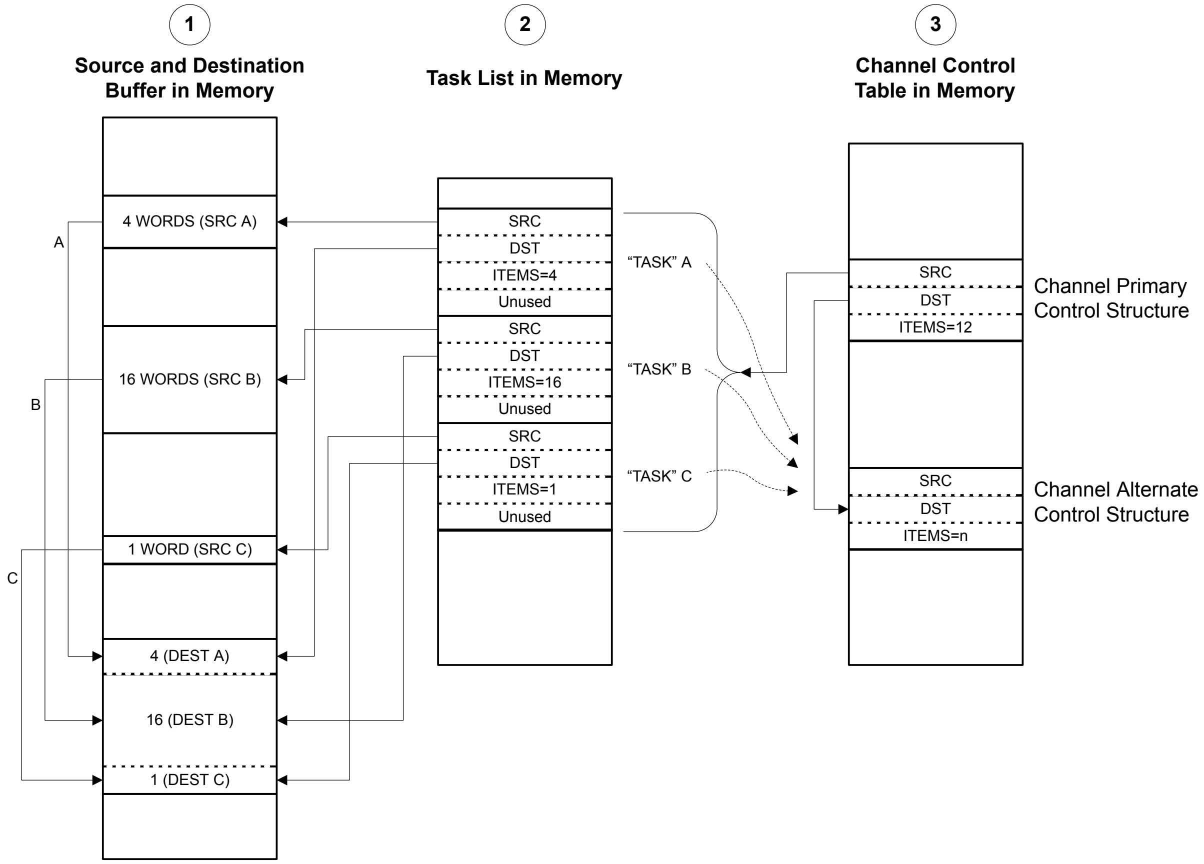
In Memory Scatter-Gather mode, the primary control structure is used to program the alternate control structure from a table in memory. The table is set up by the processor software and contains a list of control structures, each containing the source and destination end pointers, and the control word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

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

Refer to Figure 9-3 on page 236 and Figure 9-4 on page 237, which show an example of operation in Memory Scatter-Gather mode. This example shows a *gather* operation, where data in three separate buffers in memory is copied together into one buffer. Figure 9-3 on page 236 shows how the application sets up a μDMA task list in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that is used for the operation is configured to copy from the task list to the alternate control structure.

Figure 9-4 on page 237 shows the sequence as the μDMA controller performs the three sets of copy operations. First, using the primary control structure, the μDMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from the source buffer A to the destination buffer. Next, the μDMA controller again uses the primary control structure to load task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for task C.

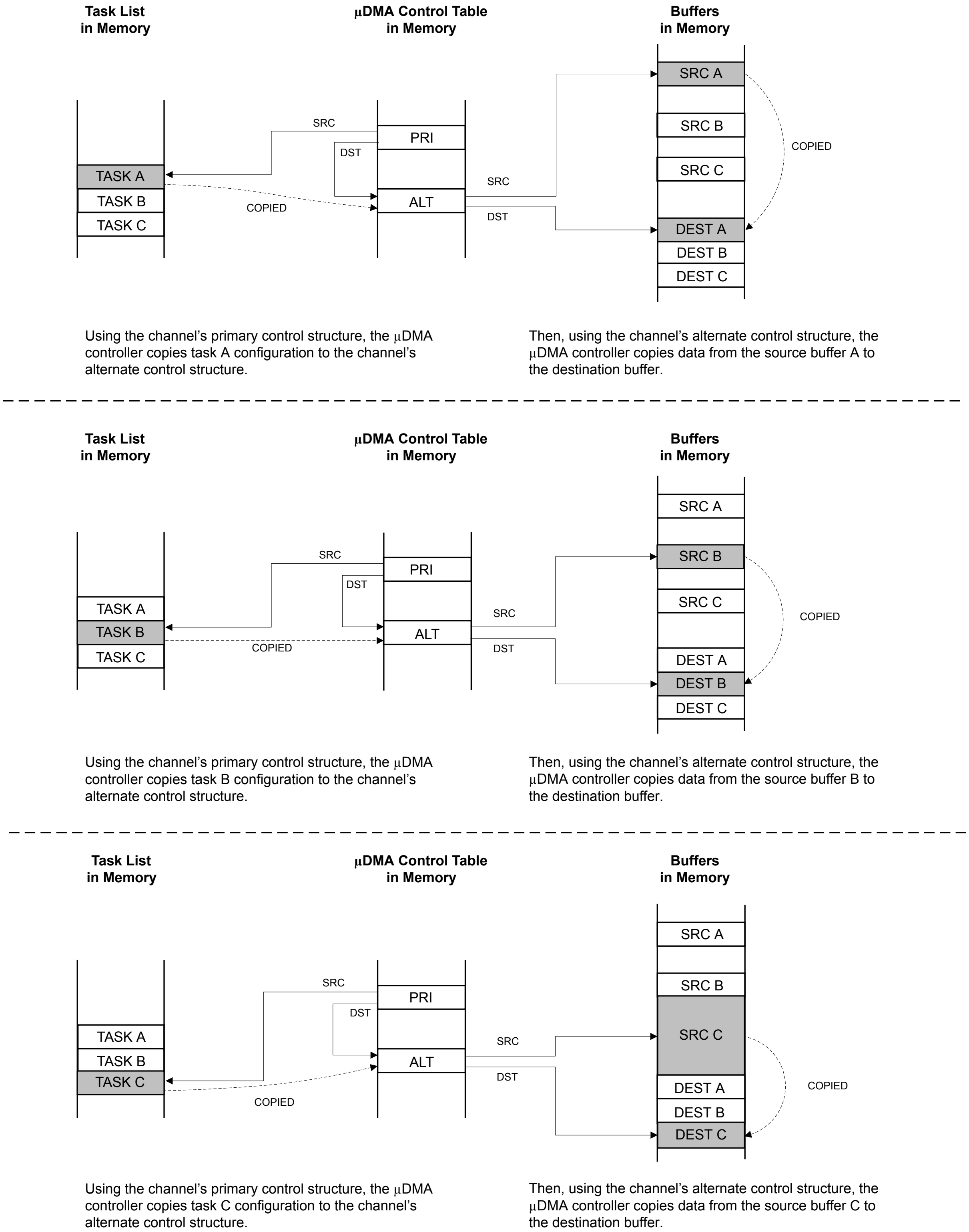
Figure 9-3. Memory Scatter-Gather, Setup and Configuration



NOTES:

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

Figure 9-4. Memory Scatter-Gather, µDMA Copy Sequence



Using the channel's primary control structure, the µDMA controller copies task A configuration to the channel's alternate control structure.

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

Using the channel's primary control structure, the µDMA controller copies task B configuration to the channel's alternate control structure.

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

Using the channel's primary control structure, the µDMA controller copies task C configuration to the channel's alternate control structure.

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

9.2.6.6 Peripheral Scatter-Gather

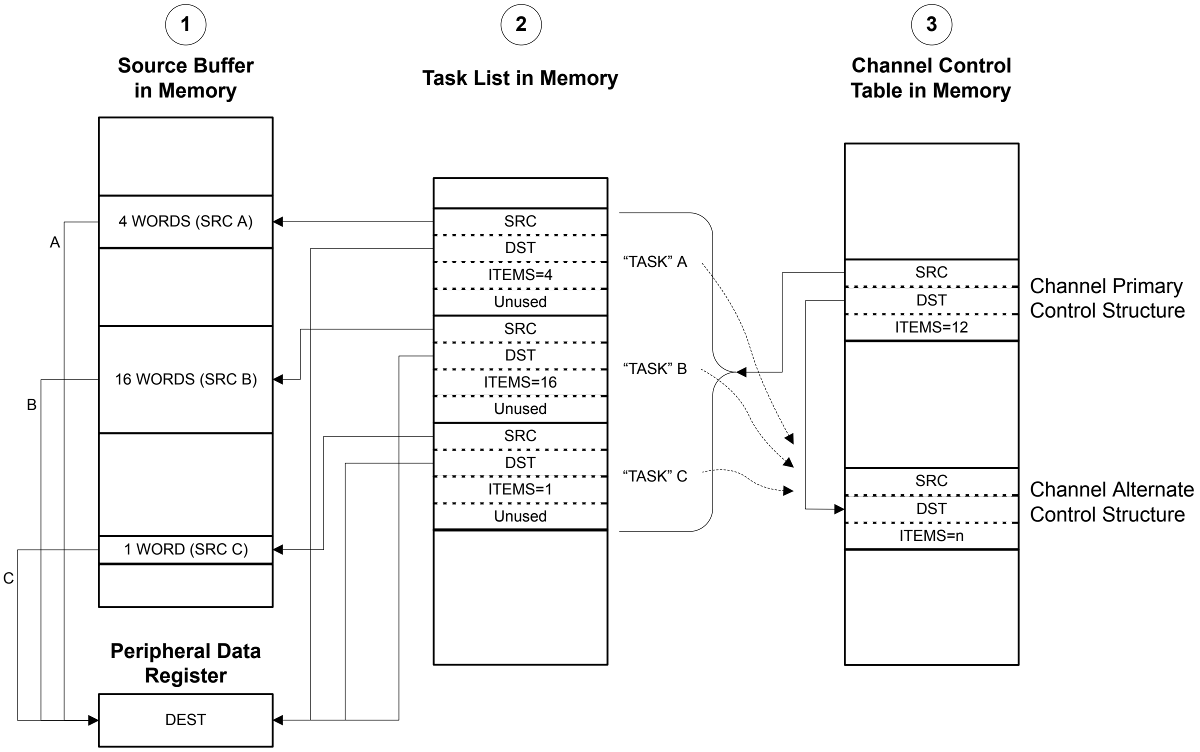
Peripheral Scatter-Gather mode is very similar to Memory Scatter-Gather, except that the transfers are controlled by a peripheral making a μDMA request. Upon detecting a request from the peripheral, the μDMA controller uses the primary control structure to copy one entry from the list to the alternate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

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

Refer to Figure 9-5 on page 239 and Figure 9-6 on page 240, which show an example of operation in Peripheral Scatter-Gather mode. This example shows a gather operation, where data from three separate buffers in memory is copied to a single peripheral data register. Figure 9-5 on page 239 shows how the application sets up a μDMA task list in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that is used for the operation is configured to copy from the task list to the alternate control structure.

Figure 9-6 on page 240 shows the sequence as the μDMA controller performs the three sets of copy operations. First, using the primary control structure, the μDMA controller loads the alternate control structure with task A. It then performs the copy operation specified by task A, copying the data from the source buffer A to the peripheral data register. Next, the μDMA controller again uses the primary control structure to load task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for task C.

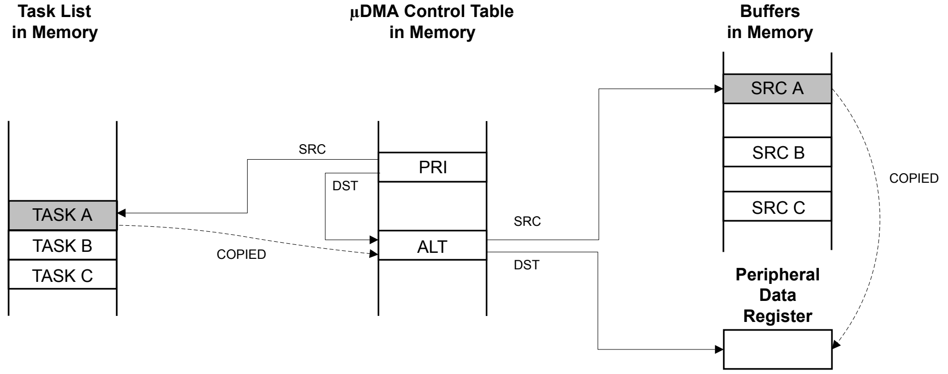
Figure 9-5. Peripheral Scatter-Gather, Setup and Configuration



NOTES:

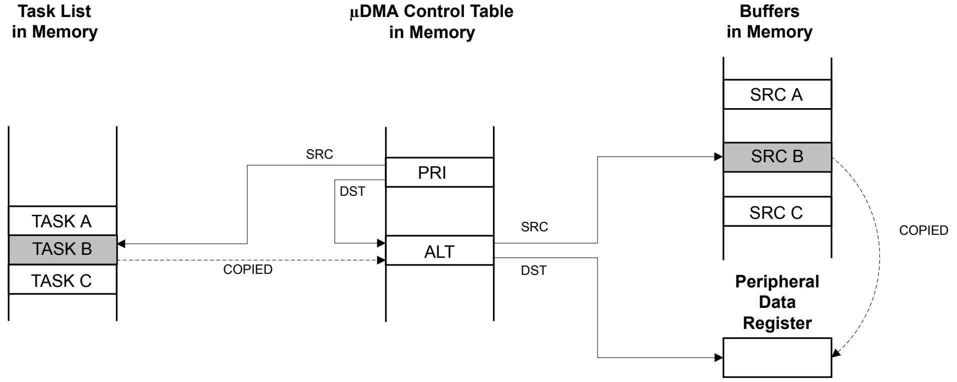
1. Application has a need to copy data items from three separate locations in memory into a peripheral data register.
2. Application sets up μDMA "task list" in memory, which contains the pointers and control configuration for three μDMA copy "tasks."
3. Application sets up the channel primary control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

Figure 9-6. Peripheral Scatter-Gather, µDMA Copy Sequence



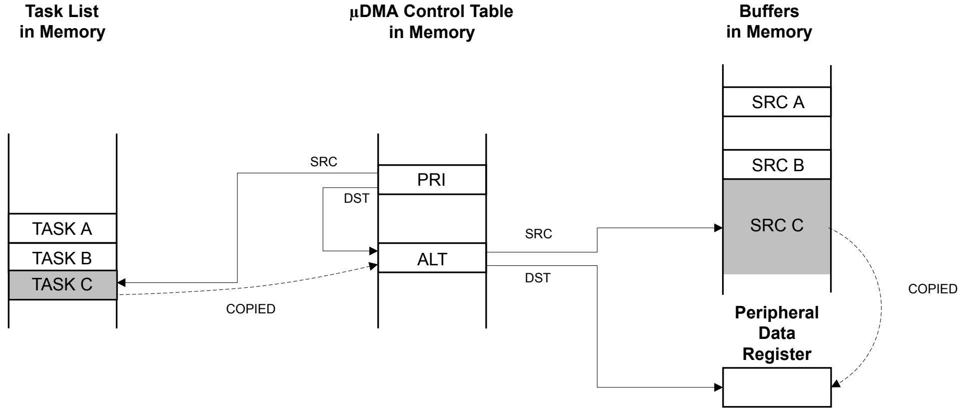
Using the channel's primary control structure, the µDMA controller copies task A configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the µDMA controller copies data from the source buffer A to the peripheral data register.



Using the channel's primary control structure, the µDMA controller copies task B configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the µDMA controller copies data from the source buffer B to the peripheral data register.



Using the channel's primary control structure, the µDMA controller copies task C configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the µDMA controller copies data from the source buffer C to the peripheral data register.

9.2.7 Transfer Size and Increment

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

Table 9-5 shows the configuration to read from a peripheral that supplies 8-bit data.

Table 9-5. µDMA Read Example: 8-Bit Peripheral

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

9.2.8 Peripheral Interface

Each peripheral that supports µDMA has a single request and/or burst request signal that is asserted when the peripheral is ready to transfer data (see Table 9-2 on page 230). The request signal can be disabled or enabled using the **DMA Channel Request Mask Set (DMAREQMASKSET)** and **DMA Channel Request Mask Clear (DMAREQMASKCLR)** registers. The µDMA request signal is disabled, or masked, when the channel request mask bit is set. When the request is not masked, the µDMA channel is configured correctly and enabled, and the peripheral asserts the request signal, the µDMA controller begins the transfer.

When a µDMA transfer is complete, the µDMA controller generates an interrupt, see “Interrupts and Errors” on page 242 for more information.

For more information on how a specific peripheral interacts with the µDMA controller, refer to the DMA Operation section in the chapter that discusses that peripheral.

9.2.9 Software Request

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

It is possible to initiate a transfer on any channel using the **DMASWREQ** register. If a request is initiated by software using a peripheral µDMA channel, then the completion interrupt occurs on the interrupt vector for the peripheral instead of the software interrupt vector. Any channel may be used for software requests as long as the corresponding peripheral is not using µDMA for data transfer.

9.2.10 Interrupts and Errors

When a μDMA transfer is complete, the μDMA controller generates a completion interrupt on the interrupt vector of the peripheral. Therefore, if μDMA is used to transfer data for a peripheral and interrupts are used, then the interrupt handler for that peripheral must be designed to handle the μDMA transfer completion interrupt. If the transfer uses the software μDMA channel, then the completion interrupt occurs on the dedicated software μDMA interrupt vector (see Table 9-6).

When μDMA is enabled for a peripheral, the μDMA controller stops the normal transfer interrupts for a peripheral from reaching the interrupt controller (the interrupts are still reported in the peripheral's interrupt registers). Thus, when a large amount of data is transferred using μDMA, instead of receiving multiple interrupts from the peripheral as data flows, the interrupt controller receives only one interrupt when the transfer is complete. Unmasked peripheral error interrupts continue to be sent to the interrupt controller.

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

Table 9-6 shows the dedicated interrupt assignments for the μDMA controller.

Table 9-6. μDMA Interrupt Assignments

Interrupt	Assignment
46	μDMA Software Channel Transfer
47	μDMA Error

9.3 Initialization and Configuration

9.3.1 Module Initialization

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

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

1. The μDMA peripheral must be enabled in the System Control block. To do this, set the `UDMA` bit of the System Control **RCGC2** register (see page 156).
2. Enable the μDMA controller by setting the `MASTEREN` bit of the **DMA Configuration (DMACFG)** register.
3. Program the location of the channel control table by writing the base address of the table to the **DMA Channel Control Base Pointer (DMACTLBASE)** register. The base address must be aligned on a 1024-byte boundary.

9.3.2 Configuring a Memory-to-Memory Transfer

μDMA channel 30 is dedicated for software-initiated transfers. However, any channel can be used for software-initiated, memory-to-memory transfer if the associated peripheral is not being used.

9.3.2.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Program bit 30 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.
2. Set bit 30 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.
3. Set bit 30 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the μDMA controller to respond to single and burst requests.
4. Set bit 30 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the μDMA controller to recognize requests for this channel.

9.3.2.2 Configure the Channel Control Structure

Now the channel control structure must be configured.

This example transfers 256 words from one memory buffer to another. Channel 30 is used for a software transfer, and the control structure for channel 30 is at offset 0x1E0 of the channel control table. The channel control structure for channel 30 is located at the offsets shown in Table 9-7.

Table 9-7. Channel Control Structure Offsets for Channel 30

Offset	Description
Control Table Base + 0x1E0	Channel 30 Source End Pointer
Control Table Base + 0x1E4	Channel 30 Destination End Pointer
Control Table Base + 0x1E8	Channel 30 Control Word

Configure the Source and Destination

The source and destination end pointers must be set to the last address for the transfer (inclusive).

1. Program the source end pointer at offset 0x1E0 to the address of the source buffer + 0x3FC.
2. Program the destination end pointer at offset 0x1E4 to the address of the destination buffer + 0x3FC.

The control word at offset 0x1E8 must be programmed according to Table 9-8.

Table 9-8. Channel Control Word Configuration for Memory Transfer Example

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	2	32-bit destination address increment
DSTSIZE	29:28	2	32-bit destination data size
SRCINC	27:26	2	32-bit source address increment
SRCSIZE	25:24	2	32-bit source data size
reserved	23:18	0	Reserved
ARBSIZE	17:14	3	Arbitrates after 8 transfers
XFERSIZE	13:4	255	Transfer 256 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	2	Use Auto-request transfer mode

9.3.2.3 Start the Transfer

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 30 of the **DMA Channel Enable Set (DMAENASET)** register.
2. Issue a transfer request by setting bit 30 of the **DMA Channel Software Request (DMASWREQ)** register.

The μDMA transfer begins. If the interrupt is enabled, then the processor is notified by interrupt when the transfer is complete. If needed, the status can be checked by reading bit 30 of the **DMAENASET** register. This bit is automatically cleared when the transfer is complete. The status can also be checked by reading the `XFERMODE` field of the channel control word at offset 0x1E8. This field is automatically cleared at the end of the transfer.

9.3.3 Configuring a Peripheral for Simple Transmit

This example configures the μDMA controller to transmit a buffer of data to a peripheral. The peripheral has a transmit FIFO with a trigger level of 4. The example peripheral uses μDMA channel 7.

9.3.3.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Configure bit 7 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.
2. Set bit 7 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.
3. Set bit 7 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the μDMA controller to respond to single and burst requests.
4. Set bit 7 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the μDMA controller to recognize requests for this channel.

9.3.3.2 Configure the Channel Control Structure

This example transfers 64 bytes from a memory buffer to the peripheral's transmit FIFO register using μDMA channel 7. The control structure for channel 7 is at offset 0x070 of the channel control table. The channel control structure for channel 7 is located at the offsets shown in Table 9-9.

Table 9-9. Channel Control Structure Offsets for Channel 7

Offset	Description
Control Table Base + 0x070	Channel 7 Source End Pointer
Control Table Base + 0x074	Channel 7 Destination End Pointer
Control Table Base + 0x078	Channel 7 Control Word

Configure the Source and Destination

The source and destination end pointers must be set to the last address for the transfer (inclusive). Because the peripheral pointer does not change, it simply points to the peripheral's data register.

1. Program the source end pointer at offset 0x070 to the address of the source buffer + 0x3F.
2. Program the destination end pointer at offset 0x074 to the address of the peripheral's transmit FIFO register.

The control word at offset 0x078 must be programmed according to Table 9-10.

Table 9-10. Channel Control Word Configuration for Peripheral Transmit Example

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	3	Destination address does not increment
DSTSIZE	29:28	0	8-bit destination data size
SRCINC	27:26	0	8-bit source address increment
SRCSIZE	25:24	0	8-bit source data size
reserved	23:18	0	Reserved
ARBSIZE	17:14	2	Arbitrates after 4 transfers
XFERSIZE	13:4	63	Transfer 64 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	1	Use Basic transfer mode

Note: In this example, it is not important if the peripheral makes a single request or a burst request. Because the peripheral has a FIFO that triggers at a level of 4, the arbitration size is set to 4. If the peripheral does make a burst request, then 4 bytes are transferred, which is what the FIFO can accommodate. If the peripheral makes a single request (if there is any space in the FIFO), then one byte is transferred at a time. If it is important to the application that transfers only be made in bursts, then the Channel Useburst `SET[7]` bit should be set in the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register.

9.3.3.3 Start the Transfer

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 7 of the **DMA Channel Enable Set (DMAENASET)** register.

The μDMA controller is now configured for transfer on channel 7. The controller makes transfers to the peripheral whenever the peripheral asserts a μDMA request. The transfers continue until the entire buffer of 64 bytes has been transferred. When that happens, the μDMA controller disables the channel and sets the `XFERMODE` field of the channel control word to 0 (Stopped). The status of the transfer can be checked by reading bit 7 of the **DMA Channel Enable Set (DMAENASET)** register. This bit is automatically cleared when the transfer is complete. The status can also be checked by reading the `XFERMODE` field of the channel control word at offset 0x078. This field is automatically cleared at the end of the transfer.

If peripheral interrupts are enabled, then the peripheral interrupt handler receives an interrupt when the entire transfer is complete.

9.3.4 Configuring a Peripheral for Ping-Pong Receive

This example configures the μDMA controller to continuously receive 8-bit data from a peripheral into a pair of 64-byte buffers. The peripheral has a receive FIFO with a trigger level of 8. The example peripheral uses μDMA channel 8.

9.3.4.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Configure bit 8 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.

2. Set bit 8 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.

3. Set bit 8 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the μDMA controller to respond to single and burst requests.

4. Set bit 8 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the μDMA controller to recognize requests for this channel.

9.3.4.2 Configure the Channel Control Structure

This example transfers bytes from the peripheral's receive FIFO register into two memory buffers of 64 bytes each. As data is received, when one buffer is full, the μDMA controller switches to use the other.

To use Ping-Pong buffering, both primary and alternate channel control structures must be used. The primary control structure for channel 8 is at offset 0x080 of the channel control table, and the alternate channel control structure is at offset 0x280. The channel control structures for channel 8 are located at the offsets shown in Table 9-11.

Table 9-11. Primary and Alternate Channel Control Structure Offsets for Channel 8

Offset	Description
Control Table Base + 0x080	Channel 8 Primary Source End Pointer
Control Table Base + 0x084	Channel 8 Primary Destination End Pointer
Control Table Base + 0x088	Channel 8 Primary Control Word
Control Table Base + 0x280	Channel 8 Alternate Source End Pointer
Control Table Base + 0x284	Channel 8 Alternate Destination End Pointer
Control Table Base + 0x288	Channel 8 Alternate Control Word

Configure the Source and Destination

The source and destination end pointers must be set to the last address for the transfer (inclusive). Because the peripheral pointer does not change, it simply points to the peripheral's data register. Both the primary and alternate sets of pointers must be configured.

1. Program the primary source end pointer at offset 0x080 to the address of the peripheral's receive buffer.

2. Program the primary destination end pointer at offset 0x084 to the address of ping-pong buffer A + 0x3F.

3. Program the alternate source end pointer at offset 0x280 to the address of the peripheral's receive buffer.

4. Program the alternate destination end pointer at offset 0x284 to the address of ping-pong buffer B + 0x3F.

The primary control word at offset 0x088 and the alternate control word at offset 0x288 are initially programmed the same way.

1. Program the primary channel control word at offset 0x088 according to Table 9-12.

2. Program the alternate channel control word at offset 0x288 according to Table 9-12.

Table 9-12. Channel Control Word Configuration for Peripheral Ping-Pong Receive Example

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	0	8-bit destination address increment
DSTSIZE	29:28	0	8-bit destination data size
SRCINC	27:26	3	Source address does not increment
SRCSIZE	25:24	0	8-bit source data size
reserved	23:18	0	Reserved
ARBSIZE	17:14	3	Arbitrates after 8 transfers
XFERSIZE	13:4	63	Transfer 64 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	3	Use Ping-Pong transfer mode

Note: In this example, it is not important if the peripheral makes a single request or a burst request. Because the peripheral has a FIFO that triggers at a level of 8, the arbitration size is set to 8. If the peripheral does make a burst request, then 8 bytes are transferred, which is what the FIFO can accommodate. If the peripheral makes a single request (if there is any data in the FIFO), then one byte is transferred at a time. If it is important to the application that transfers only be made in bursts, then the Channel Useburst `SET[8]` bit should be set in the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register.

9.3.4.3 Configure the Peripheral Interrupt

An interrupt handler should be configured when using μDMA Ping-Pong mode, it is best to use an interrupt handler. However, the Ping-Pong mode can be configured without interrupts by polling. The interrupt handler is triggered after each buffer is complete.

1. Configure and enable an interrupt handler for the peripheral.

9.3.4.4 Enable the μDMA Channel

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 8 of the **DMA Channel Enable Set (DMAENASET)** register.

9.3.4.5 Process Interrupts

The μDMA controller is now configured and enabled for transfer on channel 8. When the peripheral asserts the μDMA request signal, the μDMA controller makes transfers into buffer A using the primary channel control structure. When the primary transfer to buffer A is complete, it switches to the alternate channel control structure and makes transfers into buffer B. At the same time, the primary channel control word mode field is configured to indicate Stopped, and an interrupt is

When an interrupt is triggered, the interrupt handler must determine which buffer is complete and process the data or set a flag that the data must be processed by non-interrupt buffer processing code. Then the next buffer transfer must be set up.

In the interrupt handler:

1. Read the primary channel control word at offset 0x088 and check the `XFERMODE` field. If the field is 0, this means buffer A is complete. If buffer A is complete, then:
    a. Process the newly received data in buffer A or signal the buffer processing code that buffer A has data available.

   b. Reprogram the primary channel control word at offset 0x88 according to Table 9-12 on page 247.

2. Read the alternate channel control word at offset 0x288 and check the `XFERMODE` field. If the field is 0, this means buffer B is complete. If buffer B is complete, then:

   a. Process the newly received data in buffer B or signal the buffer processing code that buffer B has data available.

   b. Reprogram the alternate channel control word at offset 0x288 according to Table 9-12 on page 247.

9.3.5 Configuring Alternate Channels

Alternate peripherals can be assigned to each μDMA channel using the **DMACHALT** register. Each bit represents a μDMA channel. If the bit is set, then the alternate peripheral is used for the channel.

Refer to Table 9-1 on page 229 for alternate channel assignments.

For example, to use SSI1 Receive on channel 8 instead of UART0, set bit 8 of the **DMACHALT** register.

9.4 Register Map

Table 9-13 on page 248 lists the μDMA channel control structures and registers. The channel control structure shows the layout of one entry in the channel control table. The channel control table is located in system memory, and the location is determined by the application, that is, the base address is n/a (not applicable). In the table below, the offset for the channel control structures is the offset from the entry in the channel control table. See "Channel Configuration" on page 231 and Table 9-3 on page 232 for a description of how the entries in the channel control table are located in memory. The μDMA register addresses are given as a hexadecimal increment, relative to the μDMA base address of 0x400F.F000. Note that the μDMA module clock must be enabled before the registers can be programmed (see page 156).

Table 9-13. μDMA Register Map

Offset	Name	Type	Reset	Description	See page
μDMA Channel Control Structure (Offset from Channel Control Table Base)					
0x000	DMASRCENDP	R/W	-	DMA Channel Source Address End Pointer	250
0x004	DMADSTENDP	R/W	-	DMA Channel Destination Address End Pointer	251
0x008	DMACHCTL	R/W	-	DMA Channel Control Word	252
μDMA Registers (Offset from μDMA Base Address)					
0x000	DMASTAT	RO	0x001F.0000	DMA Status	257
0x004	DMACFG	WO	-	DMA Configuration	259
0x008	DMACTLBASE	R/W	0x0000.0000	DMA Channel Control Base Pointer	260
0x00C	DMAALTBASE	RO	0x0000.0200	DMA Alternate Channel Control Base Pointer	261
0x010	DMAWAITSTAT	RO	0x0000.0000	DMA Channel Wait-on-Request Status	262
0x014	DMASWREQ	WO	-	DMA Channel Software Request	263

Table 9-13. µDMA Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x018	DMAUSEBURSTSET	R/W	0x0000.0000	DMA Channel Useburst Set	264
0x01C	DMAUSEBURSTCLR	WO	-	DMA Channel Useburst Clear	265
0x020	DMAREQMASKSET	R/W	0x0000.0000	DMA Channel Request Mask Set	266
0x024	DMAREQMASKCLR	WO	-	DMA Channel Request Mask Clear	267
0x028	DMAENASET	R/W	0x0000.0000	DMA Channel Enable Set	268
0x02C	DMAENACLR	WO	-	DMA Channel Enable Clear	269
0x030	DMAALTSET	R/W	0x0000.0000	DMA Channel Primary Alternate Set	270
0x034	DMAALTCLR	WO	-	DMA Channel Primary Alternate Clear	271
0x038	DMAPRIOSET	R/W	0x0000.0000	DMA Channel Priority Set	272
0x03C	DMAPRIOCLR	WO	-	DMA Channel Priority Clear	273
0x04C	DMAERRCLR	R/W	0x0000.0000	DMA Bus Error Clear	274
0x500	DMACHALT	R/W	0x0000.0000	DMA Channel Alternate Select	275
0xFD0	DMAPeriphID4	RO	0x0000.0004	DMA Peripheral Identification 4	280
0xFE0	DMAPeriphID0	RO	0x0000.0030	DMA Peripheral Identification 0	276
0xFE4	DMAPeriphID1	RO	0x0000.00B2	DMA Peripheral Identification 1	277
0xFE8	DMAPeriphID2	RO	0x0000.000B	DMA Peripheral Identification 2	278
0xFEC	DMAPeriphID3	RO	0x0000.0000	DMA Peripheral Identification 3	279
0xFF0	DMAPCellID0	RO	0x0000.000D	DMA PrimeCell Identification 0	281
0xFF4	DMAPCellID1	RO	0x0000.00F0	DMA PrimeCell Identification 1	282
0xFF8	DMAPCellID2	RO	0x0000.0005	DMA PrimeCell Identification 2	283
0xFFC	DMAPCellID3	RO	0x0000.00B1	DMA PrimeCell Identification 3	284

9.5 µDMA Channel Control Structure

The µDMA Channel Control Structure holds the transfer settings for a µDMA channel. Each channel has two control structures, which are located in a table in system memory. Refer to “Channel Configuration” on page 231 for an explanation of the Channel Control Table and the Channel Control Structure.

The channel control structure is one entry in the channel control table. Each channel has a primary and alternate structure. The primary control structures are located at offsets 0x0, 0x10, 0x20 and so on. The alternate control structures are located at offsets 0x200, 0x210, 0x220, and so on.

Register 1: DMA Channel Source Address End Pointer (DMASRCENDP), offset 0x000

DMA Channel Source Address End Pointer (DMASRCENDP) is part of the Channel Control Structure and is used to specify the source address for a μDMA transfer.

The μDMA controller can transfer data to and from the on-chip SRAM. However, because the Flash memory and ROM are located on a separate internal bus, it is not possible to transfer data from the Flash memory or ROM with the μDMA controller.

Note: The offset specified is from the base address of the control structure in system memory, not the μDMA module base address.

DMA Channel Source Address End Pointer (DMASRCENDP)

Base n/a
Offset 0x000
Type R/W, reset -

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ADDR															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:0	ADDR	R/W	-	Source Address End Pointer This field points to the last address of the μDMA transfer source (inclusive). If the source address is not incrementing (the `SRCINC` field in the **DMACHCTL** register is 0x3), then this field points at the source location itself (such as a peripheral data register).

Register 2: DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004

DMA Channel Destination Address End Pointer (DMADSTENDP) is part of the Channel Control Structure and is used to specify the destination address for a μDMA transfer.

Note: The offset specified is from the base address of the control structure in system memory, not the μDMA module base address.

DMA Channel Destination Address End Pointer (DMADSTENDP)

Base n/a
Offset 0x004
Type R/W, reset -

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ADDR															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:0	ADDR	R/W	-	Destination Address End Pointer This field points to the last address of the μDMA transfer destination (inclusive). If the destination address is not incrementing (the `DSTINC` field in the **DMACHCTL** register is 0x3), then this field points at the destination location itself (such as a peripheral data register).

Register 3: DMA Channel Control Word (DMACHCTL), offset 0x008

DMA Channel Control Word (DMACHCTL) is part of the Channel Control Structure and is used to specify parameters of a μDMA transfer.

Note: The offset specified is from the base address of the control structure in system memory, not the μDMA module base address.

DMA Channel Control Word (DMACHCTL)

Base n/a
Offset 0x008
Type R/W, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	DSTINC		DSTSIZE		SRCINC		SRCSIZE		reserved						ARBSIZE	
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ARBSIZE		XFERSIZE										NXTUSEBURST	XFERMODE		
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:30	DSTINC	R/W	-	Destination Address Increment This field configures the destination address increment. The address increment value must be equal or greater than the value of the destination size (`DSTSIZE`).

Value	Description
0x0	Byte Increment by 8-bit locations
0x1	Half-word Increment by 16-bit locations
0x2	Word Increment by 32-bit locations
0x3	No increment Address remains set to the value of the Destination Address End Pointer (`DMADSTENDP`) for the channel

Bit/Field	Name	Type	Reset	Description
29:28	DSTSIZE	R/W	-	Destination Data Size This field configures the destination item data size. **Note:** `DSTSIZE` must be the same as `SRCSIZE`. Value Description 0x0 Byte 8-bit data size 0x1 Half-word 16-bit data size 0x2 Word 32-bit data size 0x3 Reserved
27:26	SRCINC	R/W	-	Source Address Increment This field configures the source address increment. The address increment value must be equal or greater than the value of the source size (`SRCSIZE`). Value Description 0x0 Byte Increment by 8-bit locations 0x1 Half-word Increment by 16-bit locations 0x2 Word Increment by 32-bit locations 0x3 No increment Address remains set to the value of the Source Address End Pointer (`DMASRCENDP`) for the channel
25:24	SRCSIZE	R/W	-	Source Data Size This field configures the source item data size. **Note:** `DSTSIZE` must be the same as `SRCSIZE`. Value Description 0x0 Byte 8-bit data size. 0x1 Half-word 16-bit data size. 0x2 Word 32-bit data size. 0x3 Reserved

<table>
<tr><th>Bit/Field</th><th>Name</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>23:18</td><td>reserved</td><td>R/W</td><td>-</td><td>Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.</td></tr>
<tr><td>17:14</td><td>ARBSIZE</td><td>R/W</td><td>-</td><td>Arbitration Size
This field configures the number of transfers that can occur before the μDMA controller re-arbitrates. The possible arbitration rate configurations represent powers of 2 and are shown below.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0x0</td><td>1 Transfer
Arbitrates after each μDMA transfer</td></tr>
<tr><td>0x1</td><td>2 Transfers</td></tr>
<tr><td>0x2</td><td>4 Transfers</td></tr>
<tr><td>0x3</td><td>8 Transfers</td></tr>
<tr><td>0x4</td><td>16 Transfers</td></tr>
<tr><td>0x5</td><td>32 Transfers</td></tr>
<tr><td>0x6</td><td>64 Transfers</td></tr>
<tr><td>0x7</td><td>128 Transfers</td></tr>
<tr><td>0x8</td><td>256 Transfers</td></tr>
<tr><td>0x9</td><td>512 Transfers</td></tr>
<tr><td>0xA-0xF</td><td>1024 Transfers
In this configuration, no arbitration occurs during the μDMA transfer because the maximum transfer size is 1024.</td></tr>
</table></td></tr>
<tr><td>13:4</td><td>XFERSIZE</td><td>R/W</td><td>-</td><td>Transfer Size (minus 1)
This field configures the total number of items to transfer. The value of this field is 1 less than the number to transfer (value 0 means transfer 1 item). The maximum value for this 10-bit field is 1023 which represents a transfer size of 1024 items.
The transfer size is the number of items, not the number of bytes. If the data size is 32 bits, then this value is the number of 32-bit words to transfer.
The μDMA controller updates this field immediately prior to entering the arbitration process, so it contains the number of outstanding items that is necessary to complete the μDMA cycle.</td></tr>
<tr><td>3</td><td>NXTUSEBURST</td><td>R/W</td><td>-</td><td>Next Useburst
This field controls whether the Useburst <code>SET[n]</code> bit is automatically set for the last transfer of a peripheral scatter-gather operation. Normally, for the last transfer, if the number of remaining items to transfer is less than the arbitration size, the μDMA controller uses single transfers to complete the transaction. If this bit is set, then the controller uses a burst transfer to complete the last transfer.</td></tr>
</table>

Bit/Field	Name	Type	Reset	Description
2:0	XFERMODE	R/W	-	µDMA Transfer Mode This field configures the operating mode of the µDMA cycle. Refer to “Transfer Modes” on page 233 for a detailed explanation of transfer modes. Because this register is in system RAM, it has no reset value. Therefore, this field should be initialized to 0 before the channel is enabled. Value Description 0x0 Stop 0x1 Basic 0x2 Auto-Request 0x3 Ping-Pong 0x4 Memory Scatter-Gather 0x5 Alternate Memory Scatter-Gather 0x6 Peripheral Scatter-Gather 0x7 Alternate Peripheral Scatter-Gather

XFERMODE Bit Field Values.

Stop

Channel is stopped or configuration data is invalid. No more transfers can occur.

Basic

For each trigger (whether from a peripheral or a software request), the µDMA controller performs the number of transfers specified by the `ARBSIZE` field.

Auto-Request

The initial request (software- or peripheral-initiated) is sufficient to complete the entire transfer of `XFERSIZE` items without any further requests.

Ping-Pong

This mode uses both the primary and alternate control structures for this channel. When the number of transfers specified by the `XFERSIZE` field have completed for the current control structure (primary or alternate), the µDMA controller switches to the other one. These switches continue until one of the control structures is not set to ping-pong mode. At that point, the µDMA controller stops. An interrupt is generated on completion of the transfers configured by each control structure. See “Ping-Pong” on page 233.

Memory Scatter-Gather

When using this mode, the primary control structure for the channel is configured to allow a list of operations (tasks) to be performed. The source address pointer specifies the start of a table of tasks to be copied to the alternate control structure for this channel. The `XFERMODE` field for the alternate control structure should be configured to 0x5 (Alternate memory scatter-gather) to perform the task. When the task completes, the µDMA switches back to the primary channel control structure, which then copies the next task to the alternate control structure. This process continues until the table of tasks is empty. The last task must have an `XFERMODE` value other than 0x5. Note that for continuous operation, the last task can update the primary channel control structure back to the start of the list or to another list. See “Memory Scatter-Gather” on page 234.

Alternate Memory Scatter-Gather

This value must be used in the alternate channel control data structure when the μDMA controller operates in Memory Scatter-Gather mode.

Peripheral Scatter-Gather

This value must be used in the primary channel control data structure when the μDMA controller operates in Peripheral Scatter-Gather mode. In this mode, the μDMA controller operates exactly the same as in Memory Scatter-Gather mode, except that instead of performing the number of transfers specified by the `XFERSIZE` field in the alternate control structure at one time, the μDMA controller only performs the number of transfers specified by the `ARBSIZE` field per trigger; see Basic mode for details. See “Peripheral Scatter-Gather” on page 238.

Alternate Peripheral Scatter-Gather

This value must be used in the alternate channel control data structure when the μDMA controller operates in Peripheral Scatter-Gather mode.

9.6 μDMA Register Descriptions

The register addresses given are relative to the μDMA base address of 0x400F.F000.

Register 4: DMA Status (DMASTAT), offset 0x000

The **DMA Status (DMASTAT)** register returns the status of the μDMA controller. You cannot read this register when the μDMA controller is in the reset state.

DMA Status (DMASTAT)

Base 0x400F.F000
Offset 0x000
Type RO, reset 0x001F.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved											DMACHANS				
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								STATE				reserved			MASTEN
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:21	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
20:16	DMACHANS	RO	0x1F	Available μDMA Channels Minus 1 This field contains a value equal to the number of μDMA channels the μDMA controller is configured to use, minus one. The value of 0x1F corresponds to 32 μDMA channels.
15:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:4	STATE	RO	0x0	Control State Machine Status This field shows the current status of the control state machine. Status can be one of the following. Value Description 0x0 Idle 0x1 Reading channel controller data. 0x2 Reading source end pointer. 0x3 Reading destination end pointer. 0x4 Reading source data. 0x5 Writing destination data. 0x6 Waiting for μDMA request to clear. 0x7 Writing channel controller data. 0x8 Stalled 0x9 Done 0xA-0xF Undefined
3:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
0	MASTEN	RO	0	Master Enable Status Value Description 0 The μDMA controller is disabled. 1 The μDMA controller is enabled.

Register 5: DMA Configuration (DMACFG), offset 0x004

The **DMACFG** register controls the configuration of the µDMA controller.

DMA Configuration (DMACFG)

Base 0x400F.F000
Offset 0x004
Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															MASTEN
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:1	reserved	WO	-	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	MASTEN	WO	-	Controller Master Enable Value Description 0 Disables the µDMA controller. 1 Enables µDMA controller.

Register 6: DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008

The **DMACTLBASE** register must be configured so that the base pointer points to a location in system memory.

The amount of system memory that must be assigned to the μDMA controller depends on the number of μDMA channels used and whether the alternate channel control data structure is used. See “Channel Configuration” on page 231 for details about the Channel Control Table. The base address must be aligned on a 1024-byte boundary. This register cannot be read when the μDMA controller is in the reset state.

DMA Channel Control Base Pointer (DMACTLBASE)

Base 0x400F.F000
Offset 0x008
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ADDR						reserved									
Type	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	RO	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	ADDR	R/W	0x0000.00	Channel Control Base Address This field contains the pointer to the base address of the channel control table. The base address must be 1024-byte aligned.
9:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 7: DMA Alternate Channel Control Base Pointer (DMAALTBASE), offset 0x00C

The **DMAALTBASE** register returns the base address of the alternate channel control data. This register removes the necessity for application software to calculate the base address of the alternate channel control structures. This register cannot be read when the μDMA controller is in the reset state.

DMA Alternate Channel Control Base Pointer (DMAALTBASE)

Base 0x400F.F000
Offset 0x00C
Type RO, reset 0x0000.0200

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ADDR															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	ADDR	RO	0x0000.0200	Alternate Channel Address Pointer This field provides the base address of the alternate channel control structures.

Register 8: DMA Channel Wait-on-Request Status (DMAWAITSTAT), offset 0x010

This read-only register indicates that the µDMA channel is waiting on a request. A peripheral can hold off the µDMA from performing a single request until the peripheral is ready for a burst request to enhance the µDMA performance. The use of this feature is dependent on the design of the peripheral and is not controllable by software in any way. This register cannot be read when the µDMA controller is in the reset state.

DMA Channel Wait-on-Request Status (DMAWAITSTAT)

Base 0x400F.F000
Offset 0x010
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WAITREQ[n]															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WAITREQ[n]															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	WAITREQ[n]	RO	0x0000.0000	Channel [n] Wait Status These bits provide the channel wait-on-request status. Bit 0 corresponds to channel 0. Value Description 1 The corresponding channel is waiting on a request. 0 The corresponding channel is not waiting on a request.

Register 9: DMA Channel Software Request (DMASWREQ), offset 0x014

Each bit of the **DMASWREQ** register represents the corresponding μDMA channel. Setting a bit generates a request for the specified μDMA channel.

DMA Channel Software Request (DMASWREQ)

Base 0x400F.F000
Offset 0x014
Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SWREQ[n]															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SWREQ[n]															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:0	SWREQ[n]	WO	-	Channel [n] Software Request These bits generate software requests. Bit 0 corresponds to channel 0. Value Description 1 Generate a software request for the corresponding channel. 0 No request generated. These bits are automatically cleared when the software request has been completed.

Register 10: DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018

Each bit of the **DMAUSEBURSTSET** register represents the corresponding μDMA channel. Setting a bit disables the channel's single request input from generating requests, configuring the channel to only accept burst requests. Reading the register returns the status of USEBURST.

If the amount of data to transfer is a multiple of the arbitration (burst) size, the corresponding `SET[n]` bit is cleared after completing the final transfer. If there are fewer items remaining to transfer than the arbitration (burst) size, the μDMA controller automatically clears the corresponding `SET[n]` bit, allowing the remaining items to transfer using single requests. In order to resume transfers using burst requests, the corresponding bit must be set again. A bit should not be set if the corresponding peripheral does not support the burst request model.

Refer to “Request Types” on page 230 for more details about request types.

DMA Channel Useburst Set (DMAUSEBURSTSET)

Base 0x400F.F000
Offset 0x018
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SET[n]															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SET[n]															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R/W	0x0000.0000	Channel [n] Useburst Set Value Description 0 μDMA channel [n] responds to single or burst requests. 1 μDMA channel [n] responds only to burst requests. Bit 0 corresponds to channel 0. This bit is automatically cleared as described above. A bit can also be manually cleared by setting the corresponding `CLR[n]` bit in the **DMAUSEBURSTCLR** register.

Register 11: DMA Channel Useburst Clear (DMAUSEBURSTCLR), offset 0x01C

Each bit of the **DMAUSEBURSTCLR** register represents the corresponding μDMA channel. Setting a bit clears the corresponding `SET[n]` bit in the **DMAUSEBURSTSET** register.

DMA Channel Useburst Clear (DMAUSEBURSTCLR)

Base 0x400F.F000
Offset 0x01C
Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CLR[n]															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CLR[n]															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Useburst Clear

Value	Description
0	No effect.
1	Setting a bit clears the corresponding `SET[n]` bit in the **DMAUSEBURSTSET** register meaning that μDMA channel [n] responds to single and burst requests.

Register 12: DMA Channel Request Mask Set (DMAREQMASKSET), offset 0x020

Each bit of the **DMAREQMASKSET** register represents the corresponding µDMA channel. Setting a bit disables µDMA requests for the channel. Reading the register returns the request mask status. When a µDMA channel's request is masked, that means the peripheral can no longer request µDMA transfers. The channel can then be used for software-initiated transfers.

DMA Channel Request Mask Set (DMAREQMASKSET)

Base 0x400F.F000
Offset 0x020
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SET[n]															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SET[n]															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R/W	0x0000.0000	Channel [n] Request Mask Set

Value	Description
0	The peripheral associated with channel [n] is enabled to request µDMA transfers.
1	The peripheral associated with channel [n] is not able to request µDMA transfers. Channel [n] may be used for software-initiated transfers.

Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAREQMASKCLR** register.

Register 13: DMA Channel Request Mask Clear (DMAREQMASKCLR), offset 0x024

Each bit of the **DMAREQMASKCLR** register represents the corresponding µDMA channel. Setting a bit clears the corresponding `SET[n]` bit in the **DMAREQMASKSET** register.

DMA Channel Request Mask Clear (DMAREQMASKCLR)

Base 0x400F.F000
Offset 0x024
Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CLR[n]															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CLR[n]															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Request Mask Clear

Value	Description
0	No effect.
1	Setting a bit clears the corresponding `SET[n]` bit in the **DMAREQMASKSET** register meaning that the peripheral associated with channel [n] is enabled to request µDMA transfers.

Register 14: DMA Channel Enable Set (DMAENASET), offset 0x028

Each bit of the **DMAENASET** register represents the corresponding μDMA channel. Setting a bit enables the corresponding μDMA channel. Reading the register returns the enable status of the channels. If a channel is enabled but the request mask is set (**DMAREQMASKSET**), then the channel can be used for software-initiated transfers.

DMA Channel Enable Set (DMAENASET)

Base 0x400F.F000
Offset 0x028
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SET[n]															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SET[n]															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R/W	0x0000.0000	Channel [n] Enable Set Value Description 0 μDMA Channel [n] is disabled. 1 μDMA Channel [n] is enabled. Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAENACLR** register.

Register 15: DMA Channel Enable Clear (DMAENACLR), offset 0x02C

Each bit of the **DMAENACLR** register represents the corresponding μDMA channel. Setting a bit clears the corresponding `SET[n]` bit in the **DMAENASET** register.

DMA Channel Enable Clear (DMAENACLR)

Base 0x400F.F000
Offset 0x02C
Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CLR[n]															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CLR[n]															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Clear Channel [n] Enable Clear

Value	Description
0	No effect.
1	Setting a bit clears the corresponding `SET[n]` bit in the **DMAENASET** register meaning that channel [n] is disabled for μDMA transfers.

Note: The controller disables a channel when it completes the μDMA cycle.

Register 16: DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030

Each bit of the **DMAALTSET** register represents the corresponding μDMA channel. Setting a bit configures the μDMA channel to use the alternate control data structure. Reading the register returns the status of which control data structure is in use for the corresponding μDMA channel.

DMA Channel Primary Alternate Set (DMAALTSET)

Base 0x400F.F000
Offset 0x030
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SET[n]															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SET[n]															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R/W	0x0000.0000	Channel [n] Alternate Set

Value	Description
0	μDMA channel [n] is using the primary control structure.
1	μDMA channel [n] is using the alternate control structure.

Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAALTCLR** register.

Note: For Ping-Pong and Scatter-Gather cycle types, the μDMA controller automatically sets these bits to select the alternate channel control data structure.

Register 17: DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034

Each bit of the **DMAALTCLR** register represents the corresponding μDMA channel. Setting a bit clears the corresponding `SET[n]` bit in the **DMAALTSET** register.

DMA Channel Primary Alternate Clear (DMAALTCLR)

Base 0x400F.F000
Offset 0x034
Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CLR[n]															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CLR[n]															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Alternate Clear

Value	Description
0	No effect.
1	Setting a bit clears the corresponding `SET[n]` bit in the **DMAALTSET** register meaning that channel [n] is using the primary control structure.

Note: For Ping-Pong and Scatter-Gather cycle types, the μDMA controller automatically sets these bits to select the alternate channel control data structure.

Register 18: DMA Channel Priority Set (DMAPRIOSET), offset 0x038

Each bit of the **DMAPRIOSET** register represents the corresponding μDMA channel. Setting a bit configures the μDMA channel to have a high priority level. Reading the register returns the status of the channel priority mask.

DMA Channel Priority Set (DMAPRIOSET)

Base 0x400F.F000
Offset 0x038
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	SET[n]															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SET[n]															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R/W	0x0000.0000	Channel [n] Priority Set

Value	Description
0	μDMA channel [n] is using the default priority level.
1	μDMA channel [n] is using a high priority level.

Bit 0 corresponds to channel 0. A bit can only be cleared by setting the corresponding `CLR[n]` bit in the **DMAPRIOCLR** register.

Register 19: DMA Channel Priority Clear (DMAPRIOCLR), offset 0x03C

Each bit of the **DMAPRIOCLR** register represents the corresponding μDMA channel. Setting a bit clears the corresponding `SET[n]` bit in the **DMAPRIOSET** register.

DMA Channel Priority Clear (DMAPRIOCLR)

Base 0x400F.F000
Offset 0x03C
Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	CLR[n]															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CLR[n]															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Priority Clear

Value	Description
0	No effect.
1	Setting a bit clears the corresponding `SET[n]` bit in the **DMAPRIOSET** register meaning that channel [n] is using the default priority level.

Register 20: DMA Bus Error Clear (DMAERRCLR), offset 0x04C

The **DMAERRCLR** register is used to read and clear the μDMA bus error status. The error status is set if the μDMA controller encountered a bus error while performing a transfer. If a bus error occurs on a channel, that channel is automatically disabled by the μDMA controller. The other channels are unaffected.

DMA Bus Error Clear (DMAERRCLR)

Base 0x400F.F000
Offset 0x04C
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															ERRCLR
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	ERRCLR	R/W1C	0	μDMA Bus Error Status Value Description 0 No bus error is pending. 1 A bus error is pending. This bit is cleared by writing a 1 to it.

Register 21: DMA Channel Alternate Select (DMACHALT), offset 0x500

Each bit of the **DMACHALT** register represents the corresponding μDMA channel. Setting a bit selects the alternate channel assignment as specified in Table 9-1 on page 229.

DMA Channel Alternate Select (DMACHALT)

Base 0x400F.F000
Offset 0x500
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CHALT[n]															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:0	CHALT[n]	R/W	-	Channel [n] Alternate Assignment Select

Value	Description
0	Use the primary channel assignment.
1	Use the alternate channel assignment.

Register 22: DMA Peripheral Identification 0 (DMAPeriphID0), offset 0xFE0

The **DMAPeriphIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA Peripheral Identification 0 (DMAPeriphID0)

Base 0x400F.F000
Offset 0xFE0
Type RO, reset 0x0000.0030

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x30	µDMA Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

Register 23: DMA Peripheral Identification 1 (DMAPeriphID1), offset 0xFE4

The **DMAPeriphIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA Peripheral Identification 1 (DMAPeriphID1)

Base 0x400F.F000
Offset 0xFE4
Type RO, reset 0x0000.00B2

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	1	0	1	1	0	0	1	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0xB2	μDMA Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 24: DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8

The **DMAPeriphIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA Peripheral Identification 2 (DMAPeriphID2)

Base 0x400F.F000
Offset 0xFE8
Type RO, reset 0x0000.000B

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x0B	μDMA Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 25: DMA Peripheral Identification 3 (DMAPeriphID3), offset 0xFEC

The **DMAPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

DMA Peripheral Identification 3 (DMAPeriphID3)

Base 0x400F.F000
Offset 0xFEC
Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x00	µDMA Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 26: DMA Peripheral Identification 4 (DMAPeriphID4), offset 0xFD0

The **DMAPeriphIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA Peripheral Identification 4 (DMAPeriphID4)

Base 0x400F.F000
Offset 0xFD0
Type RO, reset 0x0000.0004

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x04	µDMA Peripheral ID Register Can be used by software to identify the presence of this peripheral.

Register 27: DMA PrimeCell Identification 0 (DMAPCellID0), offset 0xFF0

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA PrimeCell Identification 0 (DMAPCellID0)

Base 0x400F.F000
Offset 0xFF0
Type RO, reset 0x0000.000D

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	μDMA PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

Register 28: DMA PrimeCell Identification 1 (DMAPCellID1), offset 0xFF4

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA PrimeCell Identification 1 (DMAPCellID1)

Base 0x400F.F000
Offset 0xFF4
Type RO, reset 0x0000.00F0

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	μDMA PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

Register 29: DMA PrimeCell Identification 2 (DMAPCellID2), offset 0xFF8

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA PrimeCell Identification 2 (DMAPCellID2)

Base 0x400F.F000
Offset 0xFF8
Type RO, reset 0x0000.0005

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

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

Register 30: DMA PrimeCell Identification 3 (DMAPCellID3), offset 0xFFC

The **DMAPCellIDn** registers are hard-coded, and the fields within the registers determine the reset values.

DMA PrimeCell Identification 3 (DMAPCellID3)

Base 0x400F.F000
Offset 0xFFC
Type RO, reset 0x0000.00B1

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	μDMA PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

10 General-Purpose Input/Outputs (GPIOs)

The GPIO module is composed of five physical GPIO blocks, each corresponding to an individual GPIO port (Port A, Port B, Port C, Port D, Port E). The GPIO module supports up to 33 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

- Up to 33 GPIOs, depending on configuration
- Highly flexible pin muxing allows use as GPIO or one of several peripheral functions
- 5-V-tolerant input/outputs
- Fast toggle capable of a change every two clock cycles
- Two means of port access: either Advanced High-Performance Bus (AHB) with better back-to-back access performance, or the legacy Advanced Peripheral Bus (APB) for backwards-compatibility with existing code
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
- Bit masking in both read and write operations through address lines
- Can be used to initiate an ADC sample sequence
- Pins configured as digital inputs are Schmitt-triggered
- Programmable control for GPIO pad configuration
  - Weak pull-up or pull-down resistors
  - 2-mA, 4-mA, and 8-mA pad drive for digital communication; up to four pads can be configured with an 18-mA pad drive for high-current applications
  - Slew rate control for the 8-mA drive
  - Open drain enables
  - Digital input enables

10.1 Signal Description

GPIO signals have alternate hardware functions. Table 10-2 on page 286 lists the GPIO pins and their analog and digital alternate functions. The `AINx` and `VREFA` analog signals are not 5-V tolerant and go through an isolation circuit before reaching their circuitry. These signals are configured by clearing the corresponding `DEN` bit in the **GPIO Digital Enable (GPIODEN)** register and setting the corresponding `AMSEL` bit in the **GPIO Analog Mode Select (GPIOAMSEL)** register. Other analog signals are 5-V tolerant and are connected directly to their circuitry (`C0-`, `C0+`, `C1-`, `C1+`). These signals are configured by clearing the `DEN` bit in the **GPIO Digital Enable (GPIODEN)** register. The digital alternate hardware functions are enabled by setting the appropriate bit in the **GPIO Alternate Function Select (GPIOAFSEL)** and **GPIODEN** registers and configuring the `PMCx` bit field in the **GPIO Port Control (GPIOPCTL)** register to the numeric enoding shown in the table below. Table entries that are shaded gray are the default values for the corresponding GPIO pin.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0) with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ puts the pins back to their default state.

Table 10-1. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	1	1	0	0	0x1
PA[5:2]	SSI0	1	1	0	0	0x1
PB[3:2]	I^{2}C0	1	1	0	0	0x1
PC[3:0]	JTAG/SWD	1	1	0	1	0x3

Table 10-2. GPIO Pins and Alternate Functions

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	10	11
PA0	17	-	U0Rx	-	-	-	-	-	-	I2C1SCL	U1Rx	-	-
PA1	18	-	U0Tx	-	-	-	-	-	-	I2C1SDA	U1Tx	-	-
PA2	19	-	SSI0Clk	-	-	-	-	-	-	-	-	-	-
PA3	20	-	SSI0Fss	-	-	-	-	-	-	-	-	-	-
PA4	21	-	SSI0Rx	-	-	-	-	-	-	-	-	-	-
PA5	22	-	SSI0Tx	-	-	-	-	-	-	-	-	-	-
PA6	25	-	I2C1SCL	CCP1	-	-	-	-	-	-	-	-	-
PA7	26	-	I2C1SDA	CCP4	-	-	-	-	CCP3	-	-	-	-
PB0	41	-	CCP0	-	-	-	U1Rx	-	-	-	-	-	-
PB1	42	-	CCP2	-	-	CCP1	U1Tx	-	-	-	-	-	-
PB2	47	-	I2C0SCL	-	-	CCP3	CCP0	-	-	-	-	-	-
PB3	27	-	I2C0SDA	-	-	-	-	-	-	-	-	-	-
PB4	58	C0-	-	-	-	U2Rx	-	-	U1Rx	-	-	-	-
PB5	57	C1-	C0o	CCP5	-	CCP0	-	CCP2	U1Tx	-	-	-	-
PB6	56	VREFA C0+	CCP1	-	C0o	-	-	CCP5	-	-	-	-	-
PB7	55	-	-	-	-	NMI	-	-	-	-	-	-	-
PC0	52	-	-	-	TCK SWCLK	-	-	-	-	-	-	-	-
PC1	51	-	-	-	TMS SWDIO	-	-	-	-	-	-	-	-
PC2	50	-	-	-	TDI	-	-	-	-	-	-	-	-
PC3	49	-	-	-	TDO SWO	-	-	-	-	-	-	-	-
PC4	11	-	CCP5	-	-	-	CCP2	CCP4	-	-	CCP1	-	-

Table 10-2. GPIO Pins and Alternate Functions *(continued)*

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	10	11
PC5	14	C1+	CCP1	C1o	C0o	-	CCP3	-	-	-	-	-	-
PC6	15	-	CCP3	-	-	-	U1Rx	CCP0	-	-	-	-	-
PC7	16	-	CCP4	-	-	CCP0	U1Tx	-	C1o	-	-	-	-
PD0	61	AIN7	-	-	-	U2Rx	U1Rx	-	-	-	-	-	-
PD1	62	AIN6	-	-	-	U2Tx	U1Tx	-	-	-	-	CCP2	-
PD2	63	AIN5	U1Rx	-	-	CCP5	-	-	-	-	-	-	-
PD3	64	AIN4	U1Tx	-	-	CCP0	-	-	-	-	-	-	-
PE0	6	AIN3	-	SSI1Clk	CCP3	-	-	-	-	-	-	-	-
PE1	5	AIN2	-	SSI1Fss	-	CCP2	-	-	-	-	-	-	-
PE2	2	AIN1	CCP4	SSI1Rx	-	-	CCP2	-	-	-	-	-	-
PE3	1	AIN0	CCP1	SSI1Tx	-	-	-	-	-	-	-	-	-
PE4	8	-	CCP3	-	-	-	U2Tx	CCP2	-	-	-	-	-

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin.

10.2 Functional Description

Each GPIO port is a separate hardware instantiation of the same physical block (see Figure 10-1 on page 288 and Figure 10-2 on page 289). The LM3S1W16 microcontroller contains five ports and thus five of these physical GPIO blocks. Note that not all pins may be implemented on every block. Some GPIO pins can function as I/O signals for the on-chip peripheral modules. For information on which GPIO pins are used for alternate hardware functions, refer to Table 19-5 on page 640.

Figure 10-1. Digital I/O Pads

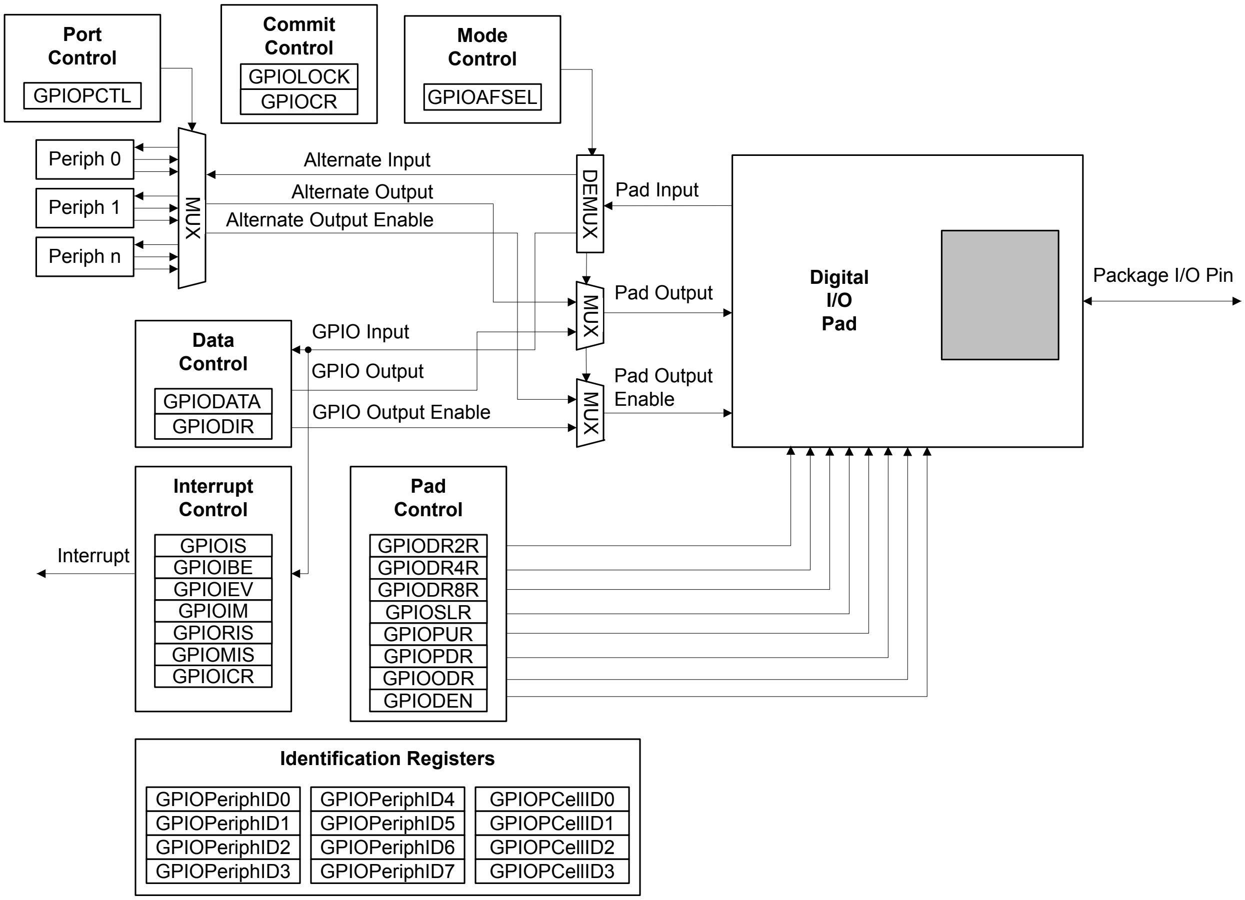
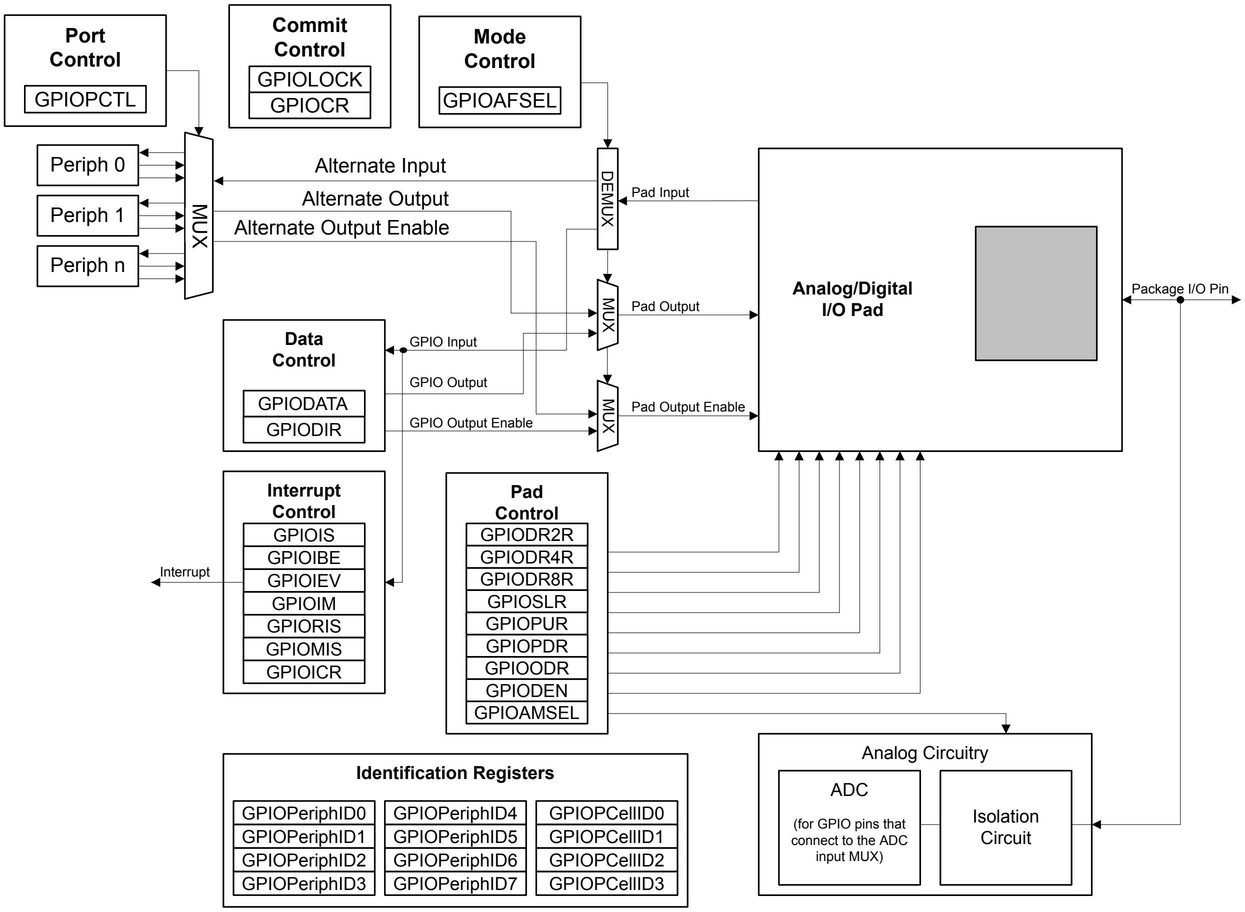


Figure 10-2. Analog/Digital I/O Pads



10.2.1 Data Control

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

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

10.2.1.1 Data Direction Operation

The **GPIO Direction (GPIODIR)** register (see page 298) is used to configure each individual pin as an input or output. When the data direction bit is cleared, the GPIO is configured as an input, and the corresponding data register bit captures and stores the value on the GPIO port. When the data direction bit is set, the GPIO is configured as an output, and the corresponding data register bit is driven out on the GPIO port.

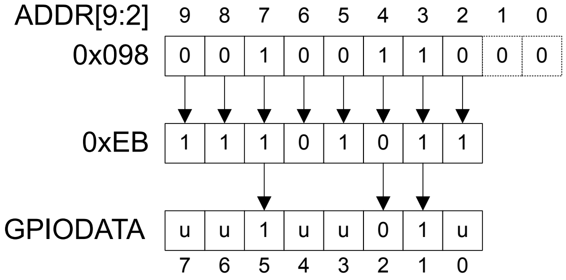
10.2.1.2 Data Register Operation

To aid in the efficiency of software, the GPIO ports allow for the modification of individual bits in the **GPIO Data (GPIODATA)** register (see page 297) by using bits [9:2] of the address bus as a mask. In this manner, software drivers can modify individual GPIO pins in a single instruction without affecting the state of the other pins. This method is more efficient than the conventional method of performing a read-modify-write operation to set or clear an individual GPIO pin. To implement this feature, the **GPIODATA** register covers 256 locations in the memory map.

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

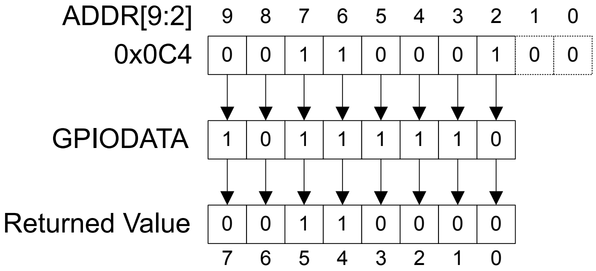
For example, writing a value of 0xEB to the address GPIODATA + 0x098 has the results shown in Figure 10-3, where u indicates that data is unchanged by the write.

Figure 10-3. GPIODATA Write Example



During a read, if the address bit associated with the data bit is set, the value is read. If the address bit associated with the data bit is cleared, the data bit is read as a zero, regardless of its actual value. For example, reading address GPIODATA + 0x0C4 yields as shown in Figure 10-4.

Figure 10-4. GPIODATA Read Example



10.2.2 Interrupt Control

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

Three registers define the edge or sense that causes interrupts:

- **GPIO Interrupt Sense (GPIOIS)** register (see page 299)

- **GPIO Interrupt Both Edges (GPIOIBE)** register (see page 300)
- **GPIO Interrupt Event (GPIOIEV)** register (see page 301)

Interrupts are enabled/disabled via the **GPIO Interrupt Mask (GPIOIM)** register (see page 302).

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 303 and page 304). As the name implies, the **GPIOMIS** register only shows interrupt conditions that are allowed to be passed to the interrupt controller. The **GPIORIS** register indicates that a GPIO pin meets the conditions for an interrupt, but has not necessarily been sent to the interrupt controller.

In addition to providing GPIO functionality, `PB4` can also be used as an external trigger for the ADC. If `PB4` is configured as a non-masked interrupt pin (the appropriate bit of `GPIOIM` is set), an interrupt for Port B is generated, and an external trigger signal is sent to the ADC. If the **ADC Event Multiplexer Select (ADCEMUX)** register is configured to use the external trigger, an ADC conversion is initiated. See page 434.

If no other Port B pins are being used to generate interrupts, the ARM Integrated Nested Vectored Interrupt Controller (NVIC) Interrupt Set Enable (SETNA) register can disable the Port B interrupts, and the ADC interrupt can be used to read back the converted data. Otherwise, the Port B interrupt handler must ignore and clear interrupts on PB4 and wait for the ADC interrupt, or the ADC interrupt must be disabled in the SETNA register and the Port B interrupt handler must poll the ADC registers until the conversion is completed. See the *ARM® Cortex™-M3 Technical Reference Manual* for more information.

Interrupts are cleared by writing a 1 to the appropriate bit of the **GPIO Interrupt Clear (GPIOICR)** register (see page 306).

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

10.2.3 Mode Control

The GPIO pins can be controlled by either software or hardware. Software control is the default for most signals and corresponds to the GPIO mode, where the **GPIODATA** register is used to read or write the corresponding pins. When hardware control is enabled via the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 307), the pin state is controlled by its alternate function (that is, the peripheral).

Further pin muxing options are provided through the **GPIO Port Control (GPIOPCTL)** register which selects one of several peripheral functions for each GPIO. For information on the configuration options, refer to Table 19-5 on page 640.

Note: If any pin is to be used as an ADC input, the appropriate bit in the **GPIOAMSEL** register must be set to disable the analog isolation circuit.

10.2.4 Commit Control

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the `NMI` pin (`PB7`) and the four JTAG/SWD pins (`PC[3:0]`). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 307), **GPIO Pull Up Select (GPIOPUR)** register (see page 313), **GPIO Pull-Down Select (GPIOPDR)** register (see page 315), and **GPIO Digital Enable (GPIODEN)** register (see page 318) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 320) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 321) have been set.

10.2.5 Pad Control

The pad control registers allow software to configure the GPIO pads based on the application requirements. The pad control registers include the **GPIODR2R**, **GPIODR4R**, **GPIODR8R**, **GPIOODR**, **GPIOPUR**, **GPIOPDR**, **GPIOSLR**, and **GPIODEN** registers. These registers control drive strength, open-drain configuration, pull-up and pull-down resistors, slew-rate control and digital input enable for each GPIO.

For special high-current applications, the GPIO output buffers may be used with the following restrictions. With the GPIO pins configured as 8-mA output drivers, a total of four GPIO outputs may be used to sink current loads up to 18 mA each. At 18-mA sink current loading, the V_{OL} value is specified as 1.2 V. The high-current GPIO package pins must be selected such that there are only a maximum of two per side of the physical package with the total number of high-current GPIO outputs not exceeding four for the entire package.

10.2.6 Identification

The identification registers configured at reset allow software to detect and identify the module as a GPIO block. The identification registers include the **GPIOPeriphID0**-**GPIOPeriphID7** registers as well as the **GPIOPCellID0**-**GPIOPCellID3** registers.

10.3 Initialization and Configuration

The GPIO modules may be accessed via two different memory apertures. The legacy aperture, the Advanced Peripheral Bus (APB), is backwards-compatible with previous Stellaris® parts. The other aperture, the Advanced High-Performance Bus (AHB), offers the same register map but provides better back-to-back access performance than the APB bus. These apertures are mutually exclusive. The aperture enabled for a given GPIO port is controlled by the appropriate bit in the **GPIOHBCTL** register (see page 110).

To use the pins in a particular GPIO port, the clock for the port must be enabled by setting the appropriate GPIO Port bit field (`GPIOn`) in the **RCGC2** register (see page 156).

On reset, all GPIO pins are configured out of reset to be undriven (tristate): **GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, and **GPIOPUR**=0, except for the pins shown in Table 10-1 on page 286. Table 10-3 on page 292 shows all possible configurations of the GPIO pads and the control register settings required to achieve them. Table 10-4 on page 293 shows how a rising edge interrupt is configured for pin 2 of a GPIO port.

Table 10-3. GPIO Pad Configuration Examples

Configuration	GPIO Register Bit Value[a]									
	AFSEL	DIR	ODR	DEN	PUR	PDR	DR2R	DR4R	DR8R	SLR
Digital Input (GPIO)	0	0	0	1	?	?	X	X	X	X
Digital Output (GPIO)	0	1	0	1	?	?	?	?	?	?
Open Drain Output (GPIO)	0	1	1	1	X	X	?	?	?	?
Open Drain Input/Output (I^2C)	1	X	1	1	X	X	?	?	?	?
Digital Input (Timer CCP)	1	X	0	1	?	?	X	X	X	X

Table 10-3. GPIO Pad Configuration Examples *(continued)*

Configuration	GPIO Register Bit Value[a]									
	AFSEL	DIR	ODR	DEN	PUR	PDR	DR2R	DR4R	DR8R	SLR
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 10-4. GPIO Interrupt Configuration Example

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

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

10.4 Register Map

Table 10-6 on page 294 lists the GPIO registers. Each GPIO port can be accessed through one of two bus apertures. The legacy aperture, the Advanced Peripheral Bus (APB), is backwards-compatible with previous Stellaris® parts. The other aperture, the Advanced High-Performance Bus (AHB), offers the same register map but provides better back-to-back access performance than the APB bus.

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

The offset listed is a hexadecimal increment to the register's address, relative to that GPIO port's base address:

- GPIO Port A (APB): 0x4000.4000
- GPIO Port A (AHB): 0x4005.8000
- GPIO Port B (APB): 0x4000.5000
- GPIO Port B (AHB): 0x4005.9000
- GPIO Port C (APB): 0x4000.6000
- GPIO Port C (AHB): 0x4005.A000
- GPIO Port D (APB): 0x4000.7000
- GPIO Port D (AHB): 0x4005.B000
- GPIO Port E (APB): 0x4002.4000
- GPIO Port E (AHB): 0x4005.C000

Note that each GPIO module clock must be enabled before the registers can be programmed (see page 156).

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0) with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ puts the pins back to their default state.

Table 10-5. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	1	1	0	0	0x1
PA[5:2]	SSI0	1	1	0	0	0x1
PB[3:2]	I^2C0	1	1	0	0	0x1
PC[3:0]	JTAG/SWD	1	1	0	1	0x3

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

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

Table 10-6. GPIO Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPIODATA	R/W	0x0000.0000	GPIO Data	297
0x400	GPIODIR	R/W	0x0000.0000	GPIO Direction	298
0x404	GPIOIS	R/W	0x0000.0000	GPIO Interrupt Sense	299

Table 10-6. GPIO Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x408	GPIOIBE	R/W	0x0000.0000	GPIO Interrupt Both Edges	300
0x40C	GPIOIEV	R/W	0x0000.0000	GPIO Interrupt Event	301
0x410	GPIOIM	R/W	0x0000.0000	GPIO Interrupt Mask	302
0x414	GPIORIS	RO	0x0000.0000	GPIO Raw Interrupt Status	303
0x418	GPIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status	304
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear	306
0x420	GPIOAFSEL	R/W	-	GPIO Alternate Function Select	307
0x500	GPIODR2R	R/W	0x0000.00FF	GPIO 2-mA Drive Select	309
0x504	GPIODR4R	R/W	0x0000.0000	GPIO 4-mA Drive Select	310
0x508	GPIODR8R	R/W	0x0000.0000	GPIO 8-mA Drive Select	311
0x50C	GPIOODR	R/W	0x0000.0000	GPIO Open Drain Select	312
0x510	GPIOPUR	R/W	-	GPIO Pull-Up Select	313
0x514	GPIOPDR	R/W	0x0000.0000	GPIO Pull-Down Select	315
0x518	GPIOSLR	R/W	0x0000.0000	GPIO Slew Rate Control Select	317
0x51C	GPIODEN	R/W	-	GPIO Digital Enable	318
0x520	GPIOLOCK	R/W	0x0000.0001	GPIO Lock	320
0x524	GPIOCR	-	-	GPIO Commit	321
0x528	GPIOAMSEL	R/W	0x0000.0000	GPIO Analog Mode Select	323
0x52C	GPIOPCTL	R/W	-	GPIO Port Control	324
0xFD0	GPIOPeriphID4	RO	0x0000.0000	GPIO Peripheral Identification 4	326
0xFD4	GPIOPeriphID5	RO	0x0000.0000	GPIO Peripheral Identification 5	327
0xFD8	GPIOPeriphID6	RO	0x0000.0000	GPIO Peripheral Identification 6	328
0xFDC	GPIOPeriphID7	RO	0x0000.0000	GPIO Peripheral Identification 7	329
0xFE0	GPIOPeriphID0	RO	0x0000.0061	GPIO Peripheral Identification 0	330
0xFE4	GPIOPeriphID1	RO	0x0000.0000	GPIO Peripheral Identification 1	331
0xFE8	GPIOPeriphID2	RO	0x0000.0018	GPIO Peripheral Identification 2	332
0xFEC	GPIOPeriphID3	RO	0x0000.0001	GPIO Peripheral Identification 3	333
0xFF0	GPIOPCellID0	RO	0x0000.000D	GPIO PrimeCell Identification 0	334
0xFF4	GPIOPCellID1	RO	0x0000.00F0	GPIO PrimeCell Identification 1	335
0xFF8	GPIOPCellID2	RO	0x0000.0005	GPIO PrimeCell Identification 2	336
0xFFC	GPIOPCellID3	RO	0x0000.00B1	GPIO PrimeCell Identification 3	337

10.5 Register Descriptions

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

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

The **GPIODATA** register is the data register. In software control mode, values written in the **GPIODATA** register are transferred onto the GPIO port pins if the respective pins have been configured as outputs through the **GPIO Direction (GPIODIR)** register (see page 298).

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

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

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

GPIO Data (GPIODATA)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0x000
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DATA							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x00	GPIO Data This register is virtually mapped to 256 locations in the address space. To facilitate the reading and writing of data to these registers by independent drivers, the data read from and written to the registers are masked by the eight address lines [9:2]. Reads from this register return its current state. Writes to this register only affect bits that are not masked by ADDR[9:2] and are configured as outputs. See "Data Register Operation" on page 290 for examples of reads and writes.

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

The **GPIODIR** register is the data direction register. Setting a bit in the **GPIODIR** register configures the corresponding pin to be an output, while clearing a bit configures the corresponding pin to be an input. All bits are cleared by a reset, meaning all GPIO pins are inputs by default.

GPIO Direction (GPIODIR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0x400
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DIR							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIR	R/W	0x00	GPIO Data Direction

Value	Description
0	Corresponding pin is an input.
1	Corresponding pins is an output.

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

The **GPIOIS** register is the interrupt sense register. Setting a bit in the **GPIOIS** register configures the corresponding pin to detect levels, while clearing a bit configures the corresponding pin to detect edges. All bits are cleared by a reset.

GPIO Interrupt Sense (GPIOIS)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0x404
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								IS							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IS	R/W	0x00	GPIO Interrupt Sense

Value	Description
0	The edge on the corresponding pin is detected (edge-sensitive).
1	The level on the corresponding pin is detected (level-sensitive).

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

The **GPIOIBE** register allows both edges to cause interrupts. When the corresponding bit in the **GPIO Interrupt Sense (GPIOIS)** register (see page 299) is set to detect edges, setting a bit in the **GPIOIBE** register configures the corresponding pin to detect both rising and falling edges, regardless of the corresponding bit in the **GPIO Interrupt Event (GPIOIEV)** register (see page 301). Clearing a bit configures the pin to be controlled by the **GPIOIEV** register. All bits are cleared by a reset.

GPIO Interrupt Both Edges (GPIOIBE)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IBE	R/W	0x00	GPIO Interrupt Both Edges

Value	Description
0	Interrupt generation is controlled by the **GPIO Interrupt Event (GPIOIEV)** register (see page 301).
1	Both edges on the corresponding pin trigger an interrupt.

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

The **GPIOIEV** register is the interrupt event register. Setting a bit in the **GPIOIEV** register configures the corresponding pin to detect rising edges or high levels, depending on the corresponding bit value in the **GPIO Interrupt Sense (GPIOIS)** register (see page 299). Clearing a bit configures the pin to detect falling edges or low levels, depending on the corresponding bit value in the **GPIOIS** register. All bits are cleared by a reset.

GPIO Interrupt Event (GPIOIEV)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0x40C
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								IEV							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IEV	R/W	0x00	GPIO Interrupt Event Value Description 0 A falling edge or a Low level on the corresponding pin triggers an interrupt. 1 A rising edge or a High level on the corresponding pin triggers an interrupt.

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

The **GPIOIM** register is the interrupt mask register. Setting a bit in the **GPIOIM** register allows interrupts that are generated by the corresponding pin to be sent to the interrupt controller on the combined interrupt signal. Clearing a bit prevents an interrupt on the corresponding pin from being sent to the interrupt controller. All bits are cleared by a reset.

GPIO Interrupt Mask (GPIOIM)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IME	R/W	0x00	GPIO Interrupt Mask Enable

Value	Description
0	The interrupt from the corresponding pin is masked.
1	The interrupt from the corresponding pin is sent to the interrupt controller.

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

The **GPIORIS** register is the raw interrupt status register. A bit in this register is set when an interrupt condition occurs on the corresponding GPIO pin. If the corresponding bit in the **GPIO Interrupt Mask (GPIOIM)** register (see page 302) is set, the interrupt is sent to the interrupt controller. Bits read as zero indicate that corresponding input pins have not initiated an interrupt. A bit in this register can be cleared by writing a 1 to the corresponding bit in the **GPIO Interrupt Clear (GPIOICR)** register.

GPIO Raw Interrupt Status (GPIORIS)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0x414
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
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	RIS	RO	0x00	GPIO Interrupt Raw Status

Value	Description
1	An interrupt condition has occurred on the corresponding pin.
0	An interrupt condition has not occurred on the corresponding pin.

A bit is cleared by writing a 1 to the corresponding bit in the **GPIOICR** register.

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

The **GPIOMIS** register is the masked interrupt status register. If a bit is set in this register, the corresponding interrupt has triggered an interrupt to the interrupt controller. If a bit is clear, either no interrupt has been generated, or the interrupt is masked.

In addition to providing GPIO functionality, `PB4` can also be used as an external trigger for the ADC. If `PB4` is configured as a non-masked interrupt pin (the appropriate bit of `GPIOIM` is set), an interrupt for Port B is generated, and an external trigger signal is sent to the ADC. If the **ADC Event Multiplexer Select (ADCEMUX)** register is configured to use the external trigger, an ADC conversion is initiated. See page 434.

If no other Port B pins are being used to generate interrupts, the ARM Integrated Nested Vectored Interrupt Controller (NVIC) Interrupt Set Enable (SETNA) register can disable the Port B interrupts, and the ADC interrupt can be used to read back the converted data. Otherwise, the Port B interrupt handler must ignore and clear interrupts on PB4 and wait for the ADC interrupt, or the ADC interrupt must be disabled in the SETNA register and the Port B interrupt handler must poll the ADC registers until the conversion is completed. See the *ARM® Cortex™-M3 Technical Reference Manual* for more information.

GPIOMIS is the state of the interrupt after masking.

GPIO Masked Interrupt Status (GPIOMIS)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0x418
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								MIS							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
7:0	MIS	RO	0x00	GPIO Masked Interrupt Status Value Description 1 An interrupt condition on the corresponding pin has triggered an interrupt to the interrupt controller. 0 An interrupt condition on the corresponding pin is masked or has not occurred. A bit is cleared by writing a 1 to the corresponding bit in the **GPIOICR** register.

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

The **GPIOICR** register is the interrupt clear register. Writing a 1 to a bit in this register clears the corresponding interrupt bit in the **GPIORIS** and **GPIOMIS** registers. Writing a 0 has no effect.

GPIO Interrupt Clear (GPIOICR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0x41C
Type W1C, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								IC							
Type	RO	RO	RO	RO	RO	RO	RO	RO	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IC	W1C	0x00	GPIO Interrupt Clear

Value	Description
1	The corresponding interrupt is cleared.
0	The corresponding interrupt is unaffected.

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

The **GPIOAFSEL** register is the mode control select register. If a bit is clear, the pin is used as a GPIO and is controlled by the GPIO registers. Setting a bit in this register configures the corresponding GPIO line to be controlled by an associated peripheral. Several possible peripheral functions are multiplexed on each GPIO. The **GPIO Port Control (GPIOPCTL)** register is used to select one of the possible functions. Table 19-5 on page 640 details which functions are muxed on each GPIO pin. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in the table below.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0) with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ puts the pins back to their default state.

Table 10-7. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	1	1	0	0	0x1
PA[5:2]	SSI0	1	1	0	0	0x1
PB[3:2]	I^{2}C0	1	1	0	0	0x1
PC[3:0]	JTAG/SWD	1	1	0	1	0x3

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

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the `NMI` pin (`PB7`) and the four JTAG/SWD pins (`PC[3:0]`). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 307), **GPIO Pull Up Select (GPIOPUR)** register (see page 313), **GPIO Pull-Down Select (GPIOPDR)** register (see page 315), and **GPIO Digital Enable (GPIODEN)** register (see page 318) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 320) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 321) have been set.

When using the I^{2}C module, in addition to setting the **GPIOAFSEL** register bits for the I^{2}C clock and data pins, the pins should be set to open drain using the **GPIO Open Drain Select (GPIOODR)** register (see examples in “Initialization and Configuration” on page 292).

GPIO Alternate Function Select (GPIOAFSEL)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	AFSEL	R/W	-	GPIO Alternate Function Select

Value	Description
0	The associated pin functions as a GPIO and is controlled by the GPIO registers.
1	The associated pin functions as a peripheral signal and is controlled by the alternate hardware function.

The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 286.

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

The **GPIODR2R** register is the 2-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the `DRV2` bit for a GPIO signal, the corresponding `DRV4` bit in the **GPIODR4R** register and `DRV8` bit in the **GPIODR8R** register are automatically cleared by hardware. By default, all GPIO pins have 2-mA drive.

GPIO 2-mA Drive Select (GPIODR2R)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV2	R/W	0xFF	Output Pad 2-mA Drive Enable

Value	Description
1	The corresponding GPIO pin has 2-mA drive.
0	The drive for the corresponding GPIO pin is controlled by the **GPIODR4R** or **GPIODR8R** register.

Setting a bit in either the **GPIODR4** register or the **GPIODR8** register clears the corresponding 2-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

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

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

GPIO 4-mA Drive Select (GPIODR4R)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0x504
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DRV4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV4	R/W	0x00	Output Pad 4-mA Drive Enable

Value	Description
1	The corresponding GPIO pin has 4-mA drive.
0	The drive for the corresponding GPIO pin is controlled by the **GPIODR2R** or **GPIODR8R** register.

Setting a bit in either the **GPIODR2** register or the **GPIODR8** register clears the corresponding 4-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

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

The **GPIODR8R** register is the 8-mA drive control register. Each GPIO signal in the port can be individually configured without affecting the other pads. When setting the `DRV8` bit for a GPIO signal, the corresponding `DRV2` bit in the **GPIODR2R** register and `DRV4` bit in the **GPIODR4R** register are automatically cleared by hardware. The 8-mA setting is also used for high-current operation.

Note: There is no configuration difference between 8-mA and high-current operation. The additional current capacity results from a shift in the V_{OH}/V_{OL} levels. See “Recommended DC Operating Conditions” on page 643 for further information.

GPIO 8-mA Drive Select (GPIODR8R)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV8	R/W	0x00	Output Pad 8-mA Drive Enable Value Description 1 The corresponding GPIO pin has 8-mA drive. 0 The drive for the corresponding GPIO pin is controlled by the **GPIODR2R** or **GPIODR4R** register. Setting a bit in either the **GPIODR2** register or the **GPIODR4** register clears the corresponding 8-mA enable bit. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

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

The **GPIOODR** register is the open drain control register. Setting a bit in this register enables the open-drain configuration of the corresponding GPIO pad. When open-drain mode is enabled, the corresponding bit should also be set in the **GPIO Digital Input Enable (GPIODEN)** register (see page 318). Corresponding bits in the drive strength and slew rate control registers (**GPIODR2R**, **GPIODR4R**, **GPIODR8R**, and **GPIOSLR**) can be set to achieve the desired rise and fall times. The GPIO acts as an open-drain input if the corresponding bit in the **GPIODIR** register is cleared. If open drain is selected while the GPIO is configured as an input, the GPIO will remain an input and the open-drain selection has no effect until the GPIO is changed to an output.

When using the I^2C module, in addition to configuring the pin to open drain, the **GPIO Alternate Function Select (GPIOAFSEL)** register bits for the I^2C clock and data pins should be set (see examples in “Initialization and Configuration” on page 292).

GPIO Open Drain Select (GPIOODR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0x50C
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								ODE							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ODE	R/W	0x00	Output Pad Open Drain Enable Value Description 1 The corresponding pin is configured as open drain. 0 The corresponding pin is not configured as open drain.

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

The **GPIOPUR** register is the pull-up control register. When a bit is set, a weak pull-up resistor on the corresponding GPIO signal is enabled. Setting a bit in **GPIOPUR** automatically clears the corresponding bit in the **GPIO Pull-Down Select (GPIOPDR)** register (see page 315). Write access to this register is protected with the **GPIOCR** register. Bits in **GPIOCR** that are cleared prevent writes to the equivalent bit in this register.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0) with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{\texttt{POR}}$) or asserting $\overline{\text{RST}}$ puts the pins back to their default state.

Table 10-8. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	1	1	0	0	0x1
PA[5:2]	SSI0	1	1	0	0	0x1
PB[3:2]	I^{2}C0	1	1	0	0	0x1
PC[3:0]	JTAG/SWD	1	1	0	1	0x3

Note: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the `NMI` pin (`PB7`) and the four JTAG/SWD pins (`PC[3:0]`). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 307), **GPIO Pull Up Select (GPIOPUR)** register (see page 313), **GPIO Pull-Down Select (GPIOPDR)** register (see page 315), and **GPIO Digital Enable (GPIODEN)** register (see page 318) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 320) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 321) have been set.

GPIO Pull-Up Select (GPIOPUR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PUE	R/W	-	Pad Weak Pull-Up Enable Value Description 1 The corresponding pin has a weak pull-up resistor. 0 The corresponding pin is not affected. Setting a bit in the **GPIOPDR** register clears the corresponding bit in the **GPIOPUR** register. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 286.

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

The **GPIOPDR** register is the pull-down control register. When a bit is set, a weak pull-down resistor on the corresponding GPIO signal is enabled. Setting a bit in **GPIOPDR** automatically clears the corresponding bit in the **GPIO Pull-Up Select (GPIOPUR)** register (see page 313).

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0) with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{\texttt{POR}}$) or asserting $\overline{\texttt{RST}}$ puts the pins back to their default state.

Table 10-9. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	1	1	0	0	0x1
PA[5:2]	SSI0	1	1	0	0	0x1
PB[3:2]	I^{2}C0	1	1	0	0	0x1
PC[3:0]	JTAG/SWD	1	1	0	1	0x3

Note: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the `NMI` pin (`PB7`) and the four JTAG/SWD pins (`PC[3:0]`). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 307), **GPIO Pull Up Select (GPIOPUR)** register (see page 313), **GPIO Pull-Down Select (GPIOPDR)** register (see page 315), and **GPIO Digital Enable (GPIODEN)** register (see page 318) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 320) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 321) have been set.

GPIO Pull-Down Select (GPIOPDR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0x514
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								PDE							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
7:0	PDE	R/W	0x00	Pad Weak Pull-Down Enable Value Description 1 The corresponding pin has a weak pull-down resistor. 0 The corresponding pin is not affected. Setting a bit in the **GPIOPUR** register clears the corresponding bit in the **GPIOPDR** register. The change is effective on the second clock cycle after the write if accessing GPIO via the APB memory aperture. If using AHB access, the change is effective on the next clock cycle.

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

The **GPIOSLR** register is the slew rate control register. Slew rate control is only available when using the 8-mA drive strength option via the **GPIO 8-mA Drive Select (GPIODR8R)** register (see page 311).

GPIO Slew Rate Control Select (GPIOSLR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0x518
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								SRL							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	SRL	R/W	0x00	Slew Rate Limit Enable (8-mA drive only)

Value	Description
1	Slew rate control is enabled for the corresponding pin.
0	Slew rate control is disabled for the corresponding pin.

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

Note: Pins configured as digital inputs are Schmitt-triggered.

The **GPIODEN** register is the digital enable register. By default, all GPIO signals except those listed below are configured out of reset to be undriven (tristate). Their digital function is disabled; they do not drive a logic value on the pin and they do not allow the pin voltage into the GPIO receiver. To use the pin as a digital input or output (either GPIO or alternate function), the corresponding `GPIODEN` bit must be set.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0) with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{\mathtt{POR}}$) or asserting $\overline{\mathtt{RST}}$ puts the pins back to their default state.

Table 10-10. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	1	1	0	0	0x1
PA[5:2]	SSI0	1	1	0	0	0x1
PB[3:2]	I^{2}C0	1	1	0	0	0x1
PC[3:0]	JTAG/SWD	1	1	0	1	0x3

Note: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the `NMI` pin (`PB7`) and the four JTAG/SWD pins (`PC[3:0]`). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 307), **GPIO Pull Up Select (GPIOPUR)** register (see page 313), **GPIO Pull-Down Select (GPIOPDR)** register (see page 315), and **GPIO Digital Enable (GPIODEN)** register (see page 318) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 320) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 321) have been set.

GPIO Digital Enable (GPIODEN)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DEN	R/W	-	Digital Enable Value Description 0 The digital functions for the corresponding pin are disabled. 1 The digital functions for the corresponding pin are enabled. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 286.

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

The **GPIOLOCK** register enables write access to the **GPIOCR** register (see page 321). Writing 0x4C4F.434B to the **GPIOLOCK** register unlocks the **GPIOCR** register. Writing any other value to the **GPIOLOCK** register re-enables the locked state. Reading the **GPIOLOCK** register returns the lock status rather than the 32-bit value that was previously written. Therefore, when write accesses are disabled, or locked, reading the **GPIOLOCK** register returns 0x0000.0001. When write accesses are enabled, or unlocked, reading the **GPIOLOCK** register returns 0x0000.0000.

GPIO Lock (GPIOLOCK)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
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	0x0000.0001	GPIO Lock A write of the value 0x4C4F.434B unlocks the **GPIO Commit (GPIOCR)** register for write access.A write of any other value or a write to the **GPIOCR** register reapplies the lock, preventing any register updates. A read of this register returns the following values:

Value	Description
0x0000.0001	The **GPIOCR** register is locked and may not be modified.
0x0000.0000	The **GPIOCR** register is unlocked and may be modified.

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

The **GPIOCR** register is the commit register. The value of the **GPIOCR** register determines which bits of the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, and **GPIODEN** registers are committed when a write to these registers is performed. If a bit in the **GPIOCR** register is cleared, the data being written to the corresponding bit in the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** registers cannot be committed and retains its previous value. If a bit in the **GPIOCR** register is set, the data being written to the corresponding bit of the **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** registers is committed to the register and reflects the new value.

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

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

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

GPIO Commit (GPIOCR)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0x524
Type -, reset -

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CR							
Type	RO	RO	RO	RO	RO	RO	RO	RO	-	-	-	-	-	-	-	-
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
7:0	CR	-	-	GPIO Commit Value Description 1 The corresponding **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** bits can be written. 0 The corresponding **GPIOAFSEL**, **GPIOPUR**, **GPIOPDR**, or **GPIODEN** bits cannot be written. **Note:** The default register type for the **GPIOCR** register is RO for all GPIO pins with the exception of the `NMI` pin and the four JTAG/SWD pins (`PB7` and `PC[3:0]`). These five pins are currently the only GPIOs that are protected by the **GPIOCR** register. Because of this, the register type for GPIO Port B7 and GPIO Port C[3:0] is R/W. The default reset value for the **GPIOCR** register is 0x0000.00FF for all GPIO pins, with the exception of the `NMI` pin and the four JTAG/SWD pins (`PB7` and `PC[3:0]`). To ensure that the JTAG port is not accidentally programmed as a GPIO, these four pins default to non-committable. To ensure that the `NMI` pin is not accidentally programmed as the non-maskable interrupt pin, it defaults to non-committable. Because of this, the default reset value of **GPIOCR** for GPIO Port B is 0x0000.007F while the default reset value of GPIOCR for Port C is 0x0000.00F0.

Register 21: GPIO Analog Mode Select (GPIOAMSEL), offset 0x528

Important: This register is only valid for ports D and E; the corresponding base addresses for the remaining ports are not valid.

If any pin is to be used as an ADC input, the appropriate bit in **GPIOAMSEL** must be set to disable the analog isolation circuit.

The **GPIOAMSEL** register controls isolation circuits to the analog side of a unified I/O pad. Because the GPIOs may be driven by a 5-V source and affect analog operation, analog circuitry requires isolation from the pins when they are not used in their analog function.

Each bit of this register controls the isolation circuitry for the corresponding GPIO signal. For information on which GPIO pins can be used for ADC functions, refer to Table 19-5 on page 640.

GPIO Analog Mode Select (GPIOAMSEL)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0x528
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												GPIOAMSEL			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	GPIOAMSEL	R/W	0x0	GPIO Analog Mode Select

Value	Description
1	The analog function of the pin is enabled, the isolation is disabled, and the pin is capable of analog functions.
0	The analog function of the pin is disabled, the isolation is enabled, and the pin is capable of digital functions as specified by the other GPIO configuration registers.

Note: This register and bits are only valid for GPIO signals that share analog function through a unified I/O pad.

The reset state of this register is 0 for all signals.

Register 22: GPIO Port Control (GPIOPCTL), offset 0x52C

The **GPIOPCTL** register is used in conjunction with the **GPIOAFSEL** register and selects the specific peripheral signal for each GPIO pin when using the alternate function mode. Most bits in the **GPIOAFSEL** register are cleared on reset, therefore most GPIO pins are configured as GPIOs by default. When a bit is set in the **GPIOAFSEL** register, the corresponding GPIO signal is controlled by an associated peripheral. The **GPIOPCTL** register selects one out of a set of peripheral functions for each GPIO, providing additional flexibility in signal definition. For information on the defined encodings for the bit fields in this register, refer to Table 19-5 on page 640. The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in the table below.

Important: All GPIO pins are configured as GPIOs and tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, **GPIOPUR**=0, and **GPIOPCTL**=0) with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ puts the pins back to their default state.

Table 10-11. GPIO Pins With Non-Zero Reset Values

GPIO Pins	Default State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL
PA[1:0]	UART0	1	1	0	0	0x1
PA[5:2]	SSI0	1	1	0	0	0x1
PB[3:2]	I^2C0	1	1	0	0	0x1
PC[3:0]	JTAG/SWD	1	1	0	1	0x3

GPIO Port Control (GPIOPCTL)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0x52C
Type R/W, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PMC7				PMC6				PMC5				PMC4			
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PMC3				PMC2				PMC1				PMC0			
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:28	PMC7	R/W	-	Port Mux Control 7 This field controls the configuration for GPIO pin 7.
27:24	PMC6	R/W	-	Port Mux Control 6 This field controls the configuration for GPIO pin 6.

Bit/Field	Name	Type	Reset	Description
23:20	PMC5	R/W	-	Port Mux Control 5 This field controls the configuration for GPIO pin 5.
19:16	PMC4	R/W	-	Port Mux Control 4 This field controls the configuration for GPIO pin 4.
15:12	PMC3	R/W	-	Port Mux Control 3 This field controls the configuration for GPIO pin 3.
11:8	PMC2	R/W	-	Port Mux Control 2 This field controls the configuration for GPIO pin 2.
7:4	PMC1	R/W	-	Port Mux Control 1 This field controls the configuration for GPIO pin 1.
3:0	PMC0	R/W	-	Port Mux Control 0 This field controls the configuration for GPIO pin 0.

Register 23: GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0

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

GPIO Peripheral Identification 4 (GPIOPeriphID4)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0xFD0
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	GPIO Peripheral ID Register [7:0]

Register 24: GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4

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

GPIO Peripheral Identification 5 (GPIOPeriphID5)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	GPIO Peripheral ID Register [15:8]

Register 25: GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8

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

GPIO Peripheral Identification 6 (GPIOPeriphID6)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0xFD8
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID6							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	GPIO Peripheral ID Register [23:16]

Register 26: GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC

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

GPIO Peripheral Identification 7 (GPIOPeriphID7)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	GPIO Peripheral ID Register [31:24]

Register 27: GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0

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

GPIO Peripheral Identification 0 (GPIOPeriphID0)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0xFE0
Type RO, reset 0x0000.0061

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x61	GPIO Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

Register 28: GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4

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

GPIO Peripheral Identification 1 (GPIOPeriphID1)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0xFE4
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								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	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	GPIO Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 29: GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8

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

GPIO Peripheral Identification 2 (GPIOPeriphID2)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0xFE8
Type RO, reset 0x0000.0018

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	GPIO Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 30: GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC

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

GPIO Peripheral Identification 3 (GPIOPeriphID3)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	GPIO Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 31: GPIO PrimeCell Identification 0 (GPIOPCellID0), offset 0xFF0

The **GPIOPCellID0**, **GPIOPCellID1**, **GPIOPCellID2**, and **GPIOPCellID3** 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 (GPIOPCellID0)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0xFF0
Type RO, reset 0x0000.000D

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	GPIO PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

Register 32: GPIO PrimeCell Identification 1 (GPIOPCellID1), offset 0xFF4

The **GPIOPCellID0**, **GPIOPCellID1**, **GPIOPCellID2**, and **GPIOPCellID3** 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 (GPIOPCellID1)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0xFF4
Type RO, reset 0x0000.00F0

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	GPIO PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

Register 33: GPIO PrimeCell Identification 2 (GPIOPCellID2), offset 0xFF8

The **GPIOPCellID0**, **GPIOPCellID1**, **GPIOPCellID2**, and **GPIOPCellID3** 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 (GPIOPCellID2)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
Offset 0xFF8
Type RO, reset 0x0000.0005

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	GPIO PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

Register 34: GPIO PrimeCell Identification 3 (GPIOPCellID3), offset 0xFFC

The **GPIOPCellID0**, **GPIOPCellID1**, **GPIOPCellID2**, and **GPIOPCellID3** 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 (GPIOPCellID3)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	GPIO PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

11 General-Purpose Timers

Programmable timers can be used to count or time external events that drive the Timer input pins. The Stellaris® General-Purpose Timer Module (GPTM) contains three GPTM blocks (Timer 0, Timer 1, and Timer 2). Each GPTM block provides two 16-bit timers/counters (referred to as Timer A and Timer B) that can be configured to operate independently as timers or event counters, or configured to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC). Timers can also be used to trigger μDMA transfers.

In addition, timers can be used to trigger analog-to-digital conversions (ADC). The ADC trigger signals from all of the general-purpose timers are ORed together before reaching the ADC module, so only one timer should be used to trigger ADC events.

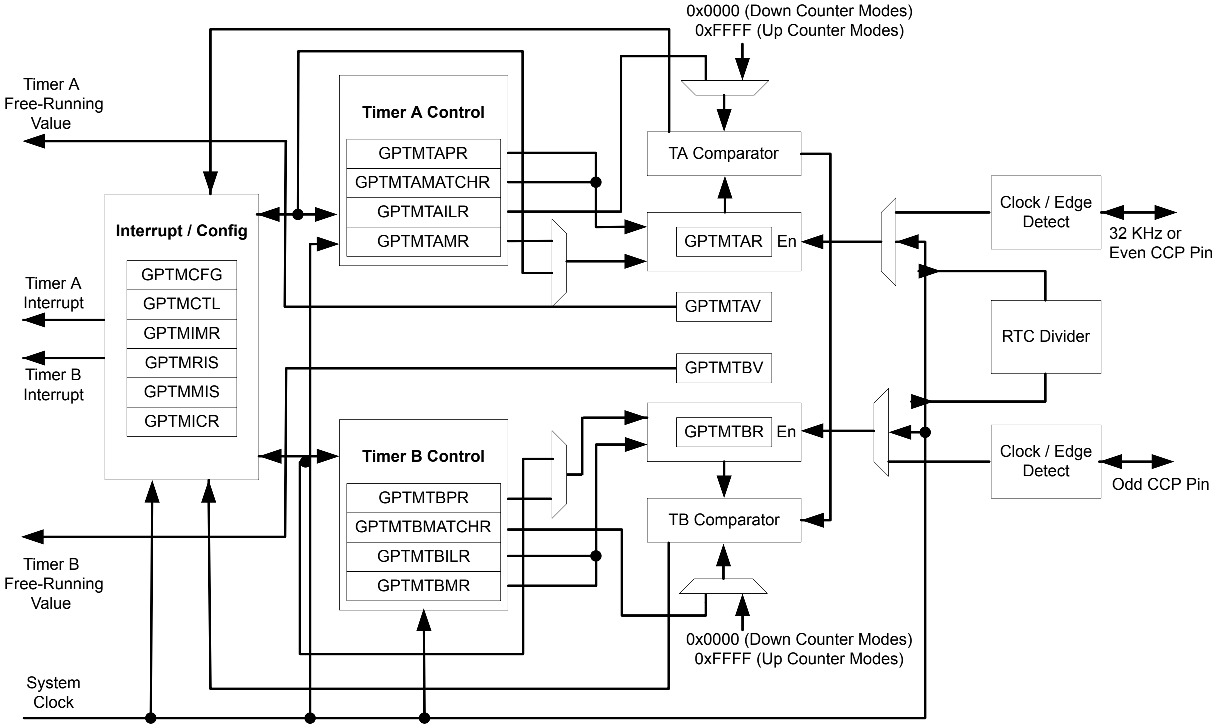
The GPT Module is one timing resource available on the Stellaris® microcontrollers. Other timer resources include the System Timer (SysTick) (see “System Timer (SysTick)” on page 57).

The General-Purpose Timer Module (GPTM) contains three GPTM blocks with the following functional options:

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

11.1 Block Diagram

Figure 11-1. GPTM Module Block Diagram



Note: In Figure 11-1 on page 339, the specific Capture Compare PWM (CCP) pins available depend on the Stellaris® device. See Table 11-1 on page 339 for the available CCP pins and their timer assignments

Table 11-1. Available CCP Pins

Timer	16-Bit Up/Down Counter	Even CCP Pin	Odd CCP Pin
Timer 0	Timer A	CCP0	-
	Timer B	-	CCP1
Timer 1	Timer A	CCP2	-
	Timer B	-	CCP3
Timer 2	Timer A	CCP4	-
	Timer B	-	CCP5

11.2 Signal Description

Table 11-2 on page 340 lists the external signals of the GP Timer module and describes the function of each. The GP Timer signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these GP Timer signals. The `AFSEL` bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 307) should be set to choose the GP Timer function. The number in parentheses is the encoding that must be programmed into the `PMCn` field in the **GPIO Port Control (GPIOPCTL)** register (page 324) to assign the GP Timer signal to the specified

GPIO port pin. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 285.

Table 11-2. Signals for General-Purpose Timers

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`CCP0`	15 16 41 47 57 64	PC6 (6) PC7 (4) PB0 (1) PB2 (5) PB5 (4) PD3 (4)	I/O	TTL	Capture/Compare/PWM 0.
`CCP1`	1 11 14 25 42 56	PE3 (1) PC4 (9) PC5 (1) PA6 (2) PB1 (4) PB6 (1)	I/O	TTL	Capture/Compare/PWM 1.
`CCP2`	2 5 8 11 42 57 62	PE2 (5) PE1 (4) PE4 (6) PC4 (5) PB1 (1) PB5 (6) PD1 (10)	I/O	TTL	Capture/Compare/PWM 2.
`CCP3`	6 8 14 15 26 47	PE0 (3) PE4 (1) PC5 (5) PC6 (1) PA7 (7) PB2 (4)	I/O	TTL	Capture/Compare/PWM 3.
`CCP4`	2 11 16 26	PE2 (1) PC4 (6) PC7 (1) PA7 (2)	I/O	TTL	Capture/Compare/PWM 4.
`CCP5`	11 56 57 63	PC4 (1) PB6 (6) PB5 (2) PD2 (4)	I/O	TTL	Capture/Compare/PWM 5.

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

11.3 Functional Description

The main components of each GPTM block are two free-running 16-bit up/down counters (referred to as Timer A and Timer B), two 16-bit match registers, 2 16-bit shadow registers, and two 16-bit load/initialization registers and their associated control functions. The exact functionality of each GPTM is controlled by software and configured through the register interface.

Software configures the GPTM using the **GPTM Configuration (GPTMCFG)** register (see page 353), the **GPTM Timer A Mode (GPTMTAMR)** register (see page 354), and the **GPTM Timer B Mode (GPTMTBMR)** register (see page 356). When in one of the 32-bit modes, the timer can only act as a 32-bit timer. However, when configured in 16-bit mode, the GPTM can have its two 16-bit timers configured in any combination of the 16-bit modes.

11.3.1 GPTM Reset Conditions

After reset has been applied to the GPTM module, the module is in an inactive state, and all control registers are cleared and in their default states. Counters Timer A and Timer B are initialized to 0xFFFF, along with their corresponding load registers: the **GPTM Timer A Interval Load (GPTMTAILR)** register (see page 371) and the **GPTM Timer B Interval Load (GPTMTBILR)** register (see page 372) and shadow registers: the **GPTM Timer A Value (GPTMTAV)** register (see page 379) and the **GPTM Timer B Value (GPTMTBV)** register (see page 380). The prescale counters are initialized to 0x00: the **GPTM Timer A Prescale (GPTMTAPR)** register (see page 375) and the **GPTM Timer B Prescale (GPTMTBPR)** register (see page 376).

11.3.2 32-Bit Timer Operating Modes

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

The GPTM is placed into 32-bit mode by writing a 0x0 (One-Shot/Periodic 32-bit timer mode) or a 0x1 (RTC mode) to the `GPTMCFG` bit field in the **GPTM Configuration (GPTMCFG)** register. In both configurations, certain GPTM registers are concatenated to form pseudo 32-bit registers. These registers include:

- **GPTM Timer A Interval Load (GPTMTAILR)** register [15:0], see page 371
- **GPTM Timer B Interval Load (GPTMTBILR)** register [15:0], see page 372
- **GPTM Timer A (GPTMTAR)** register [15:0], see page 377
- **GPTM Timer B (GPTMTBR)** register [15:0], see page 378
- **GPTM Timer A Value (GPTMTAV)** register [15:0], see page 379
- **GPTM Timer B Value (GPTMTBV)** register [15:0], see page 380

In the 32-bit modes, the GPTM translates a 32-bit write access to **GPTMTAILR** into a write access to both **GPTMTAILR** and **GPTMTBILR**. The resulting word ordering for such a write operation is:

```
GPTMTBILR[15:0]:GPTMTAILR[15:0]
```

Likewise, a 32-bit read access to **GPTMTAR** returns the value:

```
GPTMTBR[15:0]:GPTMTAR[15:0]
```

A 32-bit read access to **GPTMTAV** returns the value:

```
GPTMTBV[15:0]:GPTMTAV[15:0]
```

11.3.2.1 32-Bit One-Shot/Periodic Timer Mode

In 32-bit one-shot and periodic timer modes, the concatenated versions of the Timer A and Timer B registers are configured as a 32-bit up or down counter. The selection of one-shot or periodic mode is determined by the value written to the `TAMR` field of the **GPTM Timer A Mode (GPTMTAMR)** register (see page 354); there is no need to write to the **GPTM Timer B Mode (GPTMTBMR)** register.

When software sets the `TAEN` bit in the **GPTM Control (GPTMCTL)** register (see page 358), the timer begins counting up or down from its preloaded value. Alternatively, if the `TnWOT` bit is set in the **GPTMTnMR** register, once the `TnEN` bit is set, the timer waits for a trigger to begin counting (see "Wait-for-Trigger Mode" on page 347).

Once the time-out event (0x0000.0000 when counting down, 0xFFFF.FFFF when counting up) is reached, the timer reloads its start value from the concatenated **GPTMTAILR** on the next cycle. If configured to be a one-shot timer, the timer stops counting and clears the `TAEN` bit in the **GPTMCTL** register. If configured as a periodic timer, it continues counting. If the `TnSNAPS` bit in the **GPTMTnMR** register is set, the actual free-running value of the timer at the time-out event is loaded into the **GPTMTAR** register. In this manner, software can determine the time elapsed from the interrupt assertion to the ISR entry.

In addition to reloading the count value, the GPTM generates interrupts and triggers when it reaches the time-out event. The GPTM sets the `TATORIS` bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register (see page 363), and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register (see page 369). If the time-out interrupt is enabled in the **GPTM Interrupt Mask (GPTIMR)** register (see page 361), the GPTM also sets the `TATOMIS` bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register (see page 366). By setting the `TAMIE` bit in the **GPTMTAMR** register, an interrupt can also be generated when the Timer A value equals the value loaded into the **GPTM Timer A Match (GPTMTAMATCH)** register. This interrupt has the same status, masking, and clearing functions as the time-out interrupt. The ADC trigger is enabled by setting the `TAOTE` bit in **GPTMCTL**. The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel. See “Channel Configuration” on page 231.

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

If the `TASTALL` bit in the **GPTMCTL** register is set, the timer freezes counting until the bit is cleared.

11.3.2.2 32-Bit Real-Time Clock Timer Mode

In Real-Time Clock (RTC) mode, the concatenated versions of the Timer A and Timer B registers are configured as a 32-bit up-counter. When RTC mode is selected for the first time after reset, the counter is loaded with a value of 0x0000.0001. All subsequent load values must be written to the **GPTM Timer A Interval Load (GPTMTAILR)** register (see page 371).

The input clock on the `CCP0`, `CCP2`, or `CCP4` signal 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, the GPTM asserts the `RTCRIS` bit in **GPTMRIS** and continues counting until either a hardware reset, or it is disabled by software (clearing the `TAEN` bit). If the RTC interrupt is enabled in **GPTIMR**, the GPTM also sets the `RTCMIS` bit in **GPTMISR** and generates a controller interrupt. The status flags are cleared by writing the `RTCCINT` bit in **GPTMICR**.

In addition to generating interrupts, a µDMA trigger can be generated. The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel. See “Channel Configuration” on page 231.

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

11.3.3 16-Bit Timer Operating Modes

The GPTM is placed into global 16-bit mode by writing a value of 0x4 to the **GPTM Configuration (GPTMCFG)** register (see page 353). This section describes each of the GPTM 16-bit modes of operation. Timer A and Timer B have identical modes, so a single description is given using an **n** to reference both.

11.3.3.1 16-Bit One-Shot/Periodic Timer Mode

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

When software sets the `TnEN` bit in the **GPTMCTL** register, the timer begins counting up or down from its preloaded value. Alternatively, if the `TnWOT` bit is set in the **GPTMTnMR** register, once the `TnEN` bit is set, the timer waits for a trigger to begin counting (see "Wait-for-Trigger Mode" on page 347).

Once the time-out event (0x0000 when counting down, 0xFFFF when counting up) is reached, the timer reloads its start value from **GPTMTnILR** and **GPTMTnPR** on the next cycle. If configured to be a one-shot timer, the timer stops counting and clears the `TnEN` bit in the **GPTMCTL** register. If configured as a periodic timer, it continues counting. If the `TnSNAPS` bit in the **GPTMTnMR** register is set, the actual free-running value of the timer at the time-out event is loaded into the **GPTMTAR** register. In this manner, software can determine the time elapsed from the interrupt assertion to the ISR entry.

In addition to reloading the count value, the timer generates interrupts and triggers when it reaches the time-out event. The GPTM sets the `TnTORIS` bit in the **GPTMRIS** register, and holds it until it is cleared by writing the **GPTMICR** register. If the time-out interrupt is enabled in **GPTIMR**, the GPTM also sets the `TnTOMIS` bit in **GPTMISR** and generates a controller interrupt. By setting the `TnMIE` bit in the **GPTMTnMR** register, an interrupt can also be generated when the timer value equals the value loaded into the **GPTM Timer n Match (GPTMTnMATCH)** register. This interrupt has the same status, masking, and clearing functions as the time-out interrupt. The ADC trigger is enabled by setting the `TnOTE` bit in the **GPTMCTL** register. The µDMA trigger is enabled by configuring and enabling the appropriate µDMA channel. See "Channel Configuration" on page 231.

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

If the `TnSTALL` bit in the **GPTMCTL** register is set, the timer freezes counting until the bit is cleared.

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

Table 11-3. 16-Bit Timer With Prescaler Configurations

Prescale	#Clock (Tc)[a]	Max Time	Units
00000000	1	1.3107	mS
00000001	2	2.6214	mS
00000010	3	3.9321	mS
------------	--	--	--
11111100	254	332.9229	mS
11111110	255	334.2336	mS
11111111	256	335.5443	mS

[a] Tc is the clock period.

11.3.3.2 16-Bit Input Edge-Count Mode

Note: For rising-edge detection, the input signal must be High for at least two system clock periods following the rising edge. Similarly, for falling-edge detection, the input signal must be Low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the system frequency.

In Edge-Count mode, the timer is configured as a 16-bit down-counter with an optional 8-bit prescaler that effectively extends the counting range of the timer to 24 bits. In this mode, the timer is capable of capturing three types of events: rising edge, falling edge, or both. To place the timer in Edge-Count mode, the `TnCMR` bit of the **GPTMTnMR** register must be cleared. The type of edge that the timer counts is determined by the `TnEVENT` fields of the **GPTMCTL** register. During initialization, the **GPTM Timer n Match (GPTMTnMATCHR)** register is configured so that the difference between the value in the **GPTMTnILR** register and the **GPTMTnMATCHR** register equals the number of edge events that must be counted. The optional prescaler is loaded into the **GPTM Timer n Prescale (GPTMTnPR)** register.

When software writes the `TnEN` bit in the **GPTM Control (GPTMCTL)** register, the timer is enabled for event capture. Each input event on the `CCP` pin decrements the counter by 1 until the event count matches **GPTMTnMATCHR**. When the counts match, the GPTM asserts the `CnMRIS` bit in the **GPTMRIS** register (and the `CnMMIS` bit, if the interrupt is not masked).

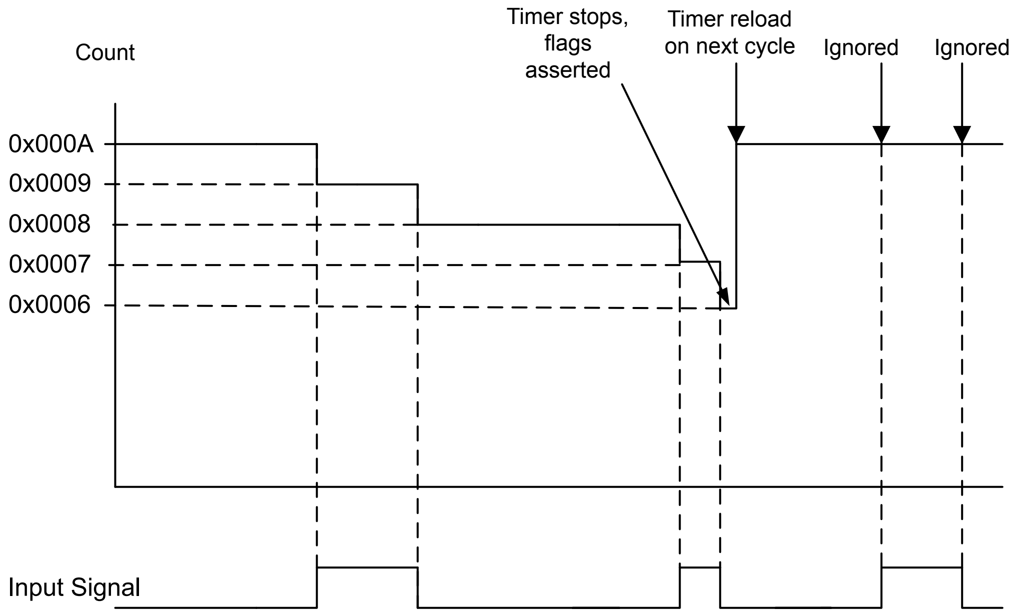
In addition to generating interrupts, a μDMA trigger can be generated. The μDMA trigger is enabled by configuring and enabling the appropriate μDMA channel. See “Channel Configuration” on page 231.

The counter is then reloaded using the value in **GPTMTnILR**, and stopped because the GPTM automatically clears the `TnEN` bit in the **GPTMCTL** register. Once the event count has been reached, all further events are ignored until `TnEN` is re-enabled by software. The **GPTMTnV** contains the free-running timer value and can be read to determine the time that elapsed between the interrupt assertion and the entry into the ISR.

Figure 11-2 on page 344 shows how Input Edge-Count mode works. In this case, the timer start value is set to **GPTMnILR** =0x000A and the match value is set to **GPTMnMATCHR** =0x0006 so that four edge events are counted. The counter is configured to detect both edges of the input signal.

Note that the last two edges are not counted since the timer automatically clears the `TnEN` bit after the current count matches the value in the **GPTMnMR** register.

Figure 11-2. 16-Bit Input Edge-Count Mode Example



11.3.3.3 16-Bit Input Edge-Time Mode

Note: For rising-edge detection, the input signal must be High for at least two system clock periods following the rising edge. Similarly, for falling edge detection, the input signal must be Low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the system frequency.

In Edge-Time mode, the timer is configured as a 16-bit free-running down-counter with an optional 8-bit prescaler that effectively extends the counting range of the timer to 24 bits. In this mode, the timer is initialized to the value loaded in the **GPTMTnILR** register (or `0xFFFF` at reset). This mode allows for event capture of either rising or falling edges, but not both. The timer is placed into Edge-Time mode by setting the `TnCMR` bit in the **GPTMTnMR** register, and the type of event that the timer captures is determined by the `TnEVENT` fields of the **GPTMCnTL** register. The optional prescaler is loaded into the **GPTM Timer n Prescale (GPTMTnPR)** register.

When software writes the `TnEN` bit in the **GPTMCTL** register, the timer is enabled for event capture. When the selected input event is detected, the current Tn counter value is captured in the **GPTMTnR** register and is available to be read by the microcontroller. The GPTM then asserts the `CnERIS` bit (and the `CnEMIS` bit, if the interrupt is not masked). The **GPTMTnV** is the free-running value of the timer and can be read to determine the time that elapsed between the interrupt assertion and the entry into the ISR.

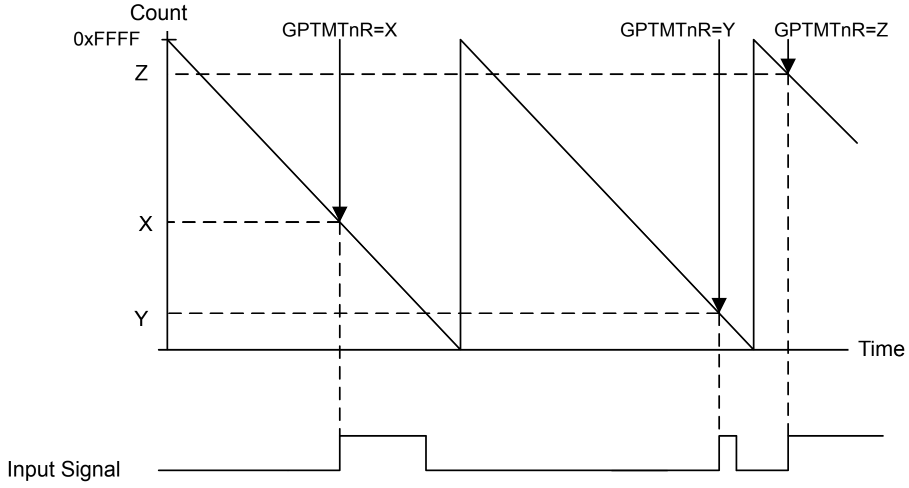
In addition to generating interrupts, a μDMA trigger can be generated. The μDMA trigger is enabled by configuring and enabling the appropriate μDMA channel. See “Channel Configuration” on page 231.

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

Figure 11-3 on page 346 shows how input edge timing mode works. In the diagram, it is assumed that the start value of the timer is the default value of `0xFFFF`, and the timer is configured to capture rising edge events.

Each time a rising edge event is detected, the current count value is loaded into the **GPTMTnR** register, and is held there until another rising edge is detected (at which point the new count value is loaded into **GPTMTnR**).

Figure 11-3. 16-Bit Input Edge-Time Mode Example



11.3.3.4 16-Bit PWM Mode

Note: The prescaler is not available in 16-Bit PWM mode.

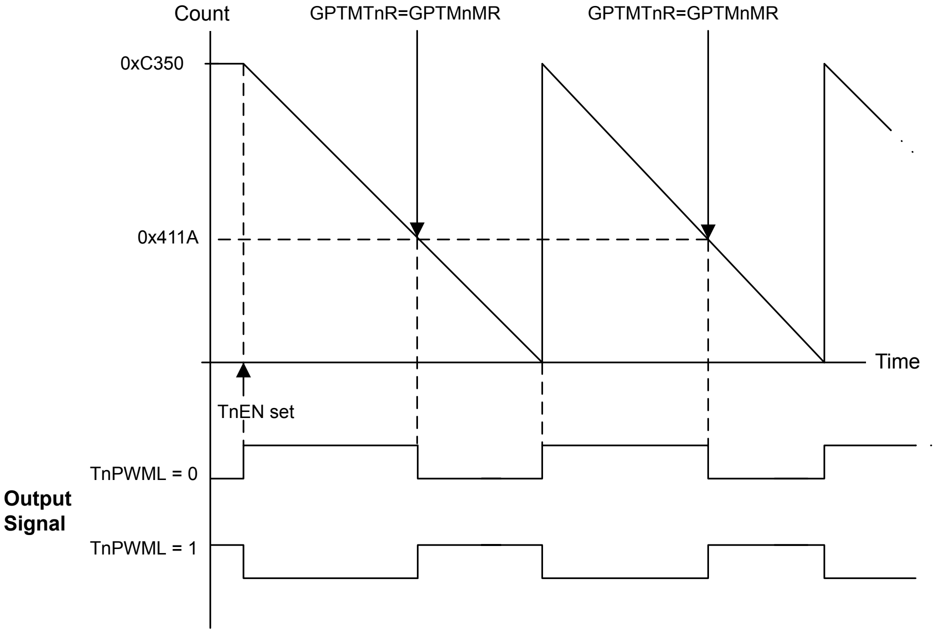
The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a down-counter with a start value (and thus period) defined by **GPTMTnILR**. PWM mode is enabled with the **GPTMTnMR** register by setting the `TnAMS` bit to `0x1`, the `TnCMR` bit to `0x0`, and the `TnMR` field to 0x2.

When software writes the `TnEN` bit in the **GPTMCTL** register, the counter begins counting down until it reaches the 0x0000 state. On the next counter cycle, the counter reloads its start value from **GPTMTnILR** and continues counting until disabled by software clearing the `TnEN` bit in the **GPTMCTL** register. No interrupts or status bits are asserted in PWM mode.

The output PWM signal asserts when the counter is at the value of the **GPTMTnILR** register (its start state), and is deasserted when the counter value equals the value in the **GPTM Timer n Match Register (GPTMnMATCHR)**. Software has the capability of inverting the output PWM signal by setting the `TnPWML` bit in the **GPTMCTL** register.

Figure 11-4 on page 347 shows how to generate an output PWM with a 1-ms period and a 66% duty cycle assuming a 50-MHz input clock and **TnPWML** =0 (duty cycle would be 33% for the **TnPWML** =1 configuration). For this example, the start value is **GPTMnIRL**=0xC350 and the match value is **GPTMnMR**=0x411A.

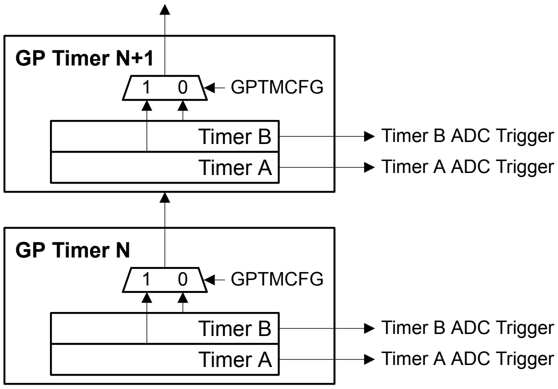
Figure 11-4. 16-Bit PWM Mode Example



11.3.3.5 Wait-for-Trigger Mode

The Wait-for-Trigger mode allows daisy chaining of the timer modules such that once configured, a single timer can initiate mulitple timing events using the Timer triggers. Wait-for-Trigger mode is enabled by setting the `TnWOT` bit in the **GPTMTnMR** register. When the `TnWOT` bit is set, Timer N+1 does not begin counting until the timer in the previous position in the daisy chain (Timer N) reaches its time-out event. The daisy chain is configured such that GPTM1 always follows GPTM0 and GPTM2 follows GPTM1. If Timer A is in 32-bit mode (controlled by the `GPTMCFG` bit in the **GPTMCFG** register), it triggers Timer A in the next module. If Timer A is in 16-bit mode, it triggers Timer B in the same module, and Timer B triggers Timer A in the next module. Care must be taken that the `TAWOT` bit is never set in GPTM0. Figure 11-5 on page 347 shows how the `GPTMCFG` bit affects the daisy chain. This function is valid for both one-shot and periodic modes.

Figure 11-5. Timer Daisy Chain



11.3.4 DMA Operation

The timers each have a dedicated μDMA channel and can provide a request signal to the μDMA controller. The request is a burst type and occurs whenever a timer raw interrupt condition occurs. The arbitration size of the μDMA transfer should be set to the amount of data that should be transferred whenever a timer event occurs.

For example, to transfer 256 items, 8 items at a time every 10 ms, configure a timer to generate a periodic timeout at 10 ms. Configure the μDMA transfer for a total of 256 items, with a burst size of 8 items. Each time the timer times out, the μDMA controller transfers 8 items, until all 256 items have been transferred.

No other special steps are needed to enable Timers for μDMA operation. Refer to “Micro Direct Memory Access (μDMA)” on page 227 for more details about programming the μDMA controller.

11.4 Initialization and Configuration

To use the general-purpose timers, the peripheral clock must be enabled by setting the `TIMER0`, `TIMER1`, and `TIMER2` bits in the **RCGC1** register (see page 147). If using any CCP pins, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module (see page 156). To find out which GPIO port to enable, refer to Table 19-4 on page 636. Configure the `PMCn` fields in the **GPIOPCTL** register to assign the CCP signals to the appropriate pins (see page 324 and Table 19-5 on page 640).

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

11.4.1 32-Bit One-Shot/Periodic Timer Mode

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

1. Ensure the timer is disabled (the `TAEN` bit in the **GPTMCTL** register is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0000.0000.
3. Configure the `TAMR` field in the **GPTM Timer A Mode Register (GPTMTAMR)**:
   a. Write a value of 0x1 for One-Shot mode.
   b. Write a value of 0x2 for Periodic mode.
4. Optionally configure the `TASNAPS`, `TAWOT`, `TAMTE`, and `TACDIR` bits in the **GPTMTAMR** register to select whether to capture the value of the free-running timer at time-out, use an external trigger to start counting, configure an additional trigger or interrupt, and count up or down.
5. Load the start value into the **GPTM Timer A Interval Load Register (GPTMTAILR)**.
6. If interrupts are required, set the appropriate bits in the **GPTM Interrupt Mask Register (GPTMIMR)**.
7. Set the `TAEN` bit in the **GPTMCTL** register to enable the timer and start counting.
8. Poll the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the appropriate bit of the **GPTM Interrupt Clear Register (GPTMICR)**.

If the `TAMIE` bit in the **GPTMTAMR** register is set, the `RTCRIS` bit in the **GPTMRIS** register is set, and the timer continues counting. In One-Shot mode, the timer stops counting after the time-out event. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode reloads the timer and continues counting after the time-out event.

11.4.2 32-Bit Real-Time Clock (RTC) Mode

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

1. Ensure the timer is disabled (the `TAEN` bit is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0000.0001.
3. Write the match value to the **GPTM Timer A Match Register (GPTMTAMATCHR)**.
4. Set/clear the `RTCEN` bit in the **GPTM Control Register (GPTMCTL)** as needed.
5. If interrupts are required, set the `RTCIM` bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.
6. Set the `TAEN` bit in the **GPTMCTL** register to enable the timer and start counting.

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

11.4.3 16-Bit One-Shot/Periodic Timer Mode

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

1. Ensure the timer is disabled (the `TnEN` bit is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0000.0004.
3. Set the `TnMR` field in the **GPTM Timer Mode (GPTMTnMR)** register:
   - **a.** Write a value of 0x1 for One-Shot mode.
   - **b.** Write a value of 0x2 for Periodic mode.
4. Optionally configure the `TnSNAPS`, `TnWOT`, `TnMTE` and `TnCDIR` bits in the **GPTMTnMR** register to select whether to capture the value of the free-running timer at time-out, use an external trigger to start counting, configure an additional trigger or interrupt, and count up or down.
5. If a prescaler is to be used, write the prescale value to the **GPTM Timer n Prescale Register (GPTMTnPR)**.
6. Load the start value into the **GPTM Timer Interval Load Register (GPTMTnILR)**.
7. If interrupts are required, set the appropriate bit in the **GPTM Interrupt Mask Register (GPTMIMR)**.
8. Set the `TnEN` bit in the **GPTM Control Register (GPTMCTL)** to enable the timer and start counting.

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

If the `TnMIE` bit in the **GPTMTnMR** register is set, the `RTCRIS` bit in the **GPTMRIS** register is set, and the timer continues counting. In One-Shot mode, the timer stops counting after the time-out event. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode reloads the timer and continues counting after the time-out event.

11.4.4 16-Bit Input Edge-Count Mode

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

1. Ensure the timer is disabled (the `TnEN` bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x0000.0004.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the `TnCMR` field to 0x0 and the `TnMR` field to 0x3.
4. Configure the type of event(s) that the timer captures by writing the `TnEVENT` field of the **GPTM Control (GPTMCTL)** register.
5. If a prescaler is to be used, write the prescale value to the **GPTM Timer n Prescale Register (GPTMTnPR)**.
6. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
7. Load the event count into the **GPTM Timer n Match (GPTMTnMATCHR)** register.
8. If interrupts are required, set the `CnMIM` bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
9. Set the `TnEN` bit in the **GPTMCTL** register to enable the timer and begin waiting for edge events.
10. Poll the `CnMRIS` bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the `CnMCINT` bit of the **GPTM Interrupt Clear (GPTMICR)** register.

In Input Edge-Count Mode, the timer stops after the programmed number of edge events has been detected. To re-enable the timer, ensure that the `TnEN` bit is cleared and repeat step 4 on page 350 through step 9 on page 350.

11.4.5 16-Bit Input Edge Timing Mode

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

1. Ensure the timer is disabled (the `TnEN` bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x0000.0004.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the `TnCMR` field to 0x1 and the `TnMR` field to 0x3.
4. Configure the type of event that the timer captures by writing the `TnEVENT` field of the **GPTM Control (GPTMCTL)** register.

5. If a prescaler is to be used, write the prescale value to the **GPTM Timer n Prescale Register (GPTMTnPR)**.
6. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
7. If interrupts are required, set the `CnEIM` bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
8. Set the `TnEN` bit in the **GPTM Control (GPTMCTL)** register to enable the timer and start counting.
9. Poll the `CnERIS` bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the `CnECINT` bit of the **GPTM Interrupt Clear (GPTMICR)** register. The time at which the event happened can be obtained by reading the **GPTM Timer n (GPTMTnR)** register.

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

11.4.6 16-Bit PWM Mode

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

1. Ensure the timer is disabled (the `TnEN` bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x0000.0004.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, set the `TnAMS` bit to 0x1, the `TnCMR` bit to 0x0, and the `TnMR` field to 0x2.
4. Configure the output state of the PWM signal (whether or not it is inverted) in the `TnEVENT` field of the **GPTM Control (GPTMCTL)** register.
5. Load the timer start value into the **GPTM Timer n Interval Load (GPTMTnILR)** register.
6. Load the **GPTM Timer n Match (GPTMTnMATCHR)** register with the match value.
7. Set the `TnEN` bit in the **GPTM Control (GPTMCTL)** register to enable the timer and begin generation of the output PWM signal.

In PWM Timing mode, the timer continues running after the PWM signal has been generated. The PWM period can be adjusted at any time by writing the **GPTMTnILR** register, and the change takes effect at the next cycle after the write.

11.5 Register Map

Table 11-4 on page 352 lists the GPTM registers. The offset listed is a hexadecimal increment to the register's address, relative to that timer's base address:

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

Note that the GP Timer module clock must be enabled before the registers can be programmed (see page 147).

Table 11-4. Timers Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPTMCFG	R/W	0x0000.0000	GPTM Configuration	353
0x004	GPTMTAMR	R/W	0x0000.0000	GPTM Timer A Mode	354
0x008	GPTMTBMR	R/W	0x0000.0000	GPTM Timer B Mode	356
0x00C	GPTMCTL	R/W	0x0000.0000	GPTM Control	358
0x018	GPTMIMR	R/W	0x0000.0000	GPTM Interrupt Mask	361
0x01C	GPTMRIS	RO	0x0000.0000	GPTM Raw Interrupt Status	363
0x020	GPTMMIS	RO	0x0000.0000	GPTM Masked Interrupt Status	366
0x024	GPTMICR	W1C	0x0000.0000	GPTM Interrupt Clear	369
0x028	GPTMTAILR	R/W	0xFFFF.FFFF	GPTM Timer A Interval Load	371
0x02C	GPTMTBILR	R/W	0x0000.FFFF	GPTM Timer B Interval Load	372
0x030	GPTMTAMATCHR	R/W	0xFFFF.FFFF	GPTM Timer A Match	373
0x034	GPTMTBMATCHR	R/W	0x0000.FFFF	GPTM Timer B Match	374
0x038	GPTMTAPR	R/W	0x0000.0000	GPTM Timer A Prescale	375
0x03C	GPTMTBPR	R/W	0x0000.0000	GPTM Timer B Prescale	376
0x048	GPTMTAR	RO	0xFFFF.FFFF	GPTM Timer A	377
0x04C	GPTMTBR	RO	0x0000.FFFF	GPTM Timer B	378
0x050	GPTMTAV	RO	0xFFFF.FFFF	GPTM Timer A Value	379
0x054	GPTMTBV	RO	0x0000.FFFF	GPTM Timer B Value	380

11.6 Register Descriptions

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

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

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

GPTM Configuration (GPTMCFG)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x000
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													GPTMCFG		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	GPTMCFG	R/W	0x0	GPTM Configuration The `GPTMCFG` values are defined as follows:

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

Register 2: GPTM Timer A Mode (GPTMTAMR), offset 0x004

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

In 16-bit timer configuration, `TAMR` controls the 16-bit timer modes for Timer A. In 32-bit timer configuration, this register controls the mode, and the contents of **GPTMTBMR** are ignored.

GPTM Timer A Mode (GPTMTAMR)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								TASNAPS	TAWOT	TAMIE	TACDIR	TAAMS	TACMR	TAMR	
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TASNAPS	R/W	0	GPTM Timer A Snap-Shot Mode Value Description 0 Snap-shot mode is disabled. 1 If Timer A is configured in the periodic mode, the actual free-running value of Timer A is loaded at the time-out event into the **GPTM Timer A (GPTMTAR)** register.
6	TAWOT	R/W	0	GPTM Timer A Wait-on-Trigger Value Description 0 Timer A begins counting as soon as it is enabled. 1 If Timer A is enabled (`TAEN` is set in the **GPTMCTL** register), Timer A does not begin counting until it receives a trigger from the timer in the previous position in the daisy chain, see Figure 11-5 on page 347. This function is valid for both one-shot and periodic modes. This bit must be clear for GP Timer Module 0, Timer A.

Bit/Field	Name	Type	Reset	Description
5	TAMIE	R/W	0	GPTM Timer A Match Interrupt Enable Value Description 0 The match interrupt is disabled. 1 An interrupt is generated when the match value in the **GPTMTAMATCHR** register is reached in the one-shot and periodic modes.
4	TACDIR	R/W	0	GPTM Timer A Count Direction Value Description 0 The timer counts down. 1 When in one-shot or periodic mode, the timer counts up. When counting up, the timer starts from a value of 0x0000.
3	TAAMS	R/W	0	GPTM Timer A Alternate Mode Select The `TAAMS` values are defined as follows: Value Description 0 Capture mode is enabled. 1 PWM mode is enabled. **Note:** To enable PWM mode, you must also clear the `TACMR` bit and configure the `TAMR` field to 0x2.
2	TACMR	R/W	0	GPTM Timer A Capture Mode The `TACMR` values are defined as follows: Value Description 0 Edge-Count mode 1 Edge-Time mode
1:0	TAMR	R/W	0x0	GPTM Timer A Mode The `TAMR` values are defined as follows: Value Description 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).

Register 3: GPTM Timer B Mode (GPTMTBMR), offset 0x008

This register configures the GPTM based on the configuration selected in the **GPTMCFG** register. When in 16-bit PWM mode, set the `TBAMS` bit, clear the `TBCMR` bit, and configure the `TBMR` field to 0x2.

In 16-bit timer configuration, these bits control the 16-bit timer modes for Timer B. In 32-bit timer configuration, this register's contents are ignored, and **GPTMTAMR** is used.

GPTM Timer B Mode (GPTMTBMR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x008
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								TBSNAPS	TBWOT	TBMIE	TBCDIR	TBAMS	TBCMR	TBMR	
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TBSNAPS	R/W	0	GPTM Timer B Snap-Shot Mode Value Description 0 Snap-shot mode is disabled. 1 If Timer B is configured in the periodic mode, the actual free-running value of Timer B is loaded at the time-out event into the **GPTM Timer B (GPTMTBR)** register.
6	TBWOT	R/W	0	GPTM Timer B Wait-on-Trigger Value Description 0 Timer B begins counting as soon as it is enabled. 1 If Timer B is enabled (`TBEN` is set in the **GPTMCTL** register), Timer B does not begin counting until it receives an it receives a trigger from the timer in the previous position in the daisy chain. See Figure 11-5 on page 347. This function is valid for both one-shot and periodic modes.
5	TBMIE	R/W	0	GPTM Timer B Match Interrupt Enable Value Description 0 The match interrupt is disabled. 1 An interrupt is generated when the match value in the **GPTMTBMATCHR** register is reached in the one-shot and periodic modes.

Bit/Field	Name	Type	Reset	Description
4	TBCDIR	R/W	0	GPTM Timer B Count Direction Value Description 0 The timer counts down. 1 When in one-shot or periodic mode, the timer counts up. When counting up, the timer starts from a value of 0x0000.
3	TBAMS	R/W	0	GPTM Timer B Alternate Mode Select The `TBAMS` values are defined as follows: Value Description 0 Capture mode is enabled. 1 PWM mode is enabled. **Note:** To enable PWM mode, you must also clear the `TBCMR` bit and set the `TBMR` field to 0x2.
2	TBCMR	R/W	0	GPTM Timer B Capture Mode The `TBCMR` values are defined as follows: Value Description 0 Edge-Count mode 1 Edge-Time mode
1:0	TBMR	R/W	0x0	GPTM Timer B Mode The `TBMR` values are defined as follows: Value Description 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.

Register 4: GPTM Control (GPTMCTL), offset 0x00C

This register is used alongside the **GPTMCFG** and **GMTMTnMR** registers to fine-tune the timer configuration, and to enable other features such as timer stall and the output trigger. The output trigger can be used to initiate transfers on the ADC module.

GPTM Control (GPTMCTL)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x00C
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	TBPWML	TBOTE	reserved	TBEVENT		TBSTALL	TBEN	reserved	TAPWML	TAOTE	RTCEN	TAEVENT		TASTALL	TAEN
Type	RO	R/W	R/W	RO	R/W	R/W	R/W	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	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	TBPWML	R/W	0	GPTM Timer B PWM Output Level The `TBPWML` values are defined as follows: Value Description 0 Output is unaffected. 1 Output is inverted.
13	TBOTE	R/W	0	GPTM Timer B Output Trigger Enable The `TBOTE` values are defined as follows: Value Description 0 The output Timer B ADC trigger is disabled. 1 The output Timer B ADC trigger is enabled. In addition, the ADC must be enabled and the timer selected as a trigger source with the `EMn` bit in the **ADCEMUX** register (see page 434).
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
11:10	TBEVENT	R/W	0x0	GPTM Timer B Event Mode The `TBEVENT` values are defined as follows: Value Description 0x0 Positive edge 0x1 Negative edge 0x2 Reserved 0x3 Both edges
9	TBSTALL	R/W	0	GPTM Timer B Stall Enable The `TBSTALL` values are defined as follows: Value Description 0 Timer B stalling is disabled. 1 Timer B stalling is enabled.
8	TBEN	R/W	0	GPTM Timer B Enable The `TBEN` values are defined as follows: Value Description 0 Timer B is disabled. 1 Timer B is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	TAPWML	R/W	0	GPTM Timer A PWM Output Level The `TAPWML` values are defined as follows: Value Description 0 Output is unaffected. 1 Output is inverted.
5	TAOTE	R/W	0	GPTM Timer A Output Trigger Enable The `TAOTE` values are defined as follows: Value Description 0 The output Timer A ADC trigger is disabled. 1 The output Timer A ADC trigger is enabled. In addition, the ADC must be enabled and the timer selected as a trigger source with the `EMn` bit in the **ADCEMUX** register (see page 434).

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

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

This register allows software to enable/disable GPTM controller-level interrupts. Setting a bit enables the corresponding interrupt, while clearing a bit disables it.

GPTM Interrupt Mask (GPTMIMR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x018
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				TBMIM	CBEIM	CBMIM	TBTOIM	reserved			TAMIM	RTCIM	CAEIM	CAMIM	TATOIM
Type	RO	RO	RO	RO	R/W	R/W	R/W	R/W	RO	RO	RO	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TBMIM	R/W	0	GPTM Timer B Mode Match Interrupt Mask The `TBMIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
10	CBEIM	R/W	0	GPTM Capture B Event Interrupt Mask The `CBEIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
9	CBMIM	R/W	0	GPTM Capture B Match Interrupt Mask The `CBMIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.

Bit/Field	Name	Type	Reset	Description
8	TBTOIM	R/W	0	GPTM Timer B Time-Out Interrupt Mask The `TBTOIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	TAMIM	R/W	0	GPTM Timer A Mode Match Interrupt Mask The `TAMIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
3	RTCIM	R/W	0	GPTM RTC Interrupt Mask The `RTCIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
2	CAEIM	R/W	0	GPTM Capture A Event Interrupt Mask The `CAEIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
1	CAMIM	R/W	0	GPTM Capture A Match Interrupt Mask The `CAMIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
0	TATOIM	R/W	0	GPTM Timer A Time-Out Interrupt Mask The `TATOIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.

Register 6: GPTM Raw Interrupt Status (GPTMRIS), offset 0x01C

This register shows the state of the GPTM's internal interrupt signal. These bits are set whether or not the interrupt is masked in the **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
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				TBMRIS	CBERIS	CBMRIS	TBTORIS	reserved			TAMRIS	RTCRIS	CAERIS	CAMRIS	TATORIS
Type	RO	RO	RO	RO	R/W	RO	RO	RO	RO	RO	RO	R/W	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TBMRIS	R/W	0	GPTM Timer B Mode Match Raw Interrupt Value Description 1 The `TBMIE` bit is set in the **GPTMTBMR** register, and the match value in the **GPTMTBMATCHR** register has been reached when in the one-shot and periodic modes. 0 The match value has not been reached. This bit is cleared by writing a 1 to the `TBMCINT` bit in the **GPTMICR** register.
10	CBERIS	RO	0	GPTM Capture B Event Raw Interrupt Value Description 1 The Capture B event has occurred. 0 The Capture B event has not occurred. This bit is cleared by writing a 1 to the `CBECINT` bit in the **GPTMICR** register.
9	CBMRIS	RO	0	GPTM Capture B Match Raw Interrupt Value Description 1 The Capture B match has occurred. 0 The Capture B match has not occurred. This bit is cleared by writing a 1 to the `CBMCINT` bit in the **GPTMICR** register.

Bit/Field	Name	Type	Reset	Description
8	TBTORIS	RO	0	GPTM Timer B Time-Out Raw Interrupt Value Description 1 Timer B has timed out. 0 Timer B has not timed out. This bit is cleared by writing a 1 to the `TBTOCINT` bit in the **GPTMICR** register.
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	TAMRIS	R/W	0	GPTM Timer A Mode Match Raw Interrupt Value Description 1 The `TAMIE` bit is set in the **GPTMTAMR** register, and the match value in the **GPTMTAMATCHR** register has been reached when in the one-shot and periodic modes. 0 The match value has not been reached. This bit is cleared by writing a 1 to the `TAMCINT` bit in the **GPTMICR** register.
3	RTCRIS	RO	0	GPTM RTC Raw Interrupt Value Description 1 The RTC event has occurred. 0 The RTC event has not occurred. This bit is cleared by writing a 1 to the `RTCCINT` bit in the **GPTMICR** register.
2	CAERIS	RO	0	GPTM Capture A Event Raw Interrupt Value Description 1 The Capture A event has occurred. 0 The Capture A event has not occurred. This bit is cleared by writing a 1 to the `CAECINT` bit in the **GPTMICR** register.
1	CAMRIS	RO	0	GPTM Capture A Match Raw Interrupt Value Description 1 The Capture A match has occurred. 0 The Capture A match has not occurred. This bit is cleared by writing a 1 to the `CAMCINT` bit in the **GPTMICR** register.

Bit/Field	Name	Type	Reset	Description
0	TATORIS	RO	0	GPTM Timer A Time-Out Raw Interrupt Value Description 1 Timer A has timed out. 0 Timer A has not timed out. This bit is cleared by writing a 1 to the `TATOCINT` bit in the **GPTMICR** register.

Register 7: GPTM Masked Interrupt Status (GPTMMIS), offset 0x020

This register 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
Offset 0x020
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				TBMMIS	CBEMIS	CBMMIS	TBTOMIS	reserved			TAMMIS	RTCMIS	CAEMIS	CAMMIS	TATOMIS
Type	RO	RO	RO	RO	R/W	RO	RO	RO	RO	RO	RO	R/W	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TBMMIS	R/W	0	GPTM Timer B Mode Match Masked Interrupt Value Description 1 An unmasked Timer B Mode Match interrupt has occurred. 0 A Timer B Mode Match interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `TBMCINT` bit in the **GPTMICR** register.
10	CBEMIS	RO	0	GPTM Capture B Event Masked Interrupt Value Description 1 An unmasked Capture B event interrupt has occurred. 0 A Capture B event interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `CBECINT` bit in the **GPTMICR** register.
9	CBMMIS	RO	0	GPTM Capture B Match Masked Interrupt Value Description 1 An unmasked Capture B Match interrupt has occurred. 0 A Capture B Mode Match interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `CBMCINT` bit in the **GPTMICR** register.

Bit/Field	Name	Type	Reset	Description
8	TBTOMIS	RO	0	GPTM Timer B Time-Out Masked Interrupt Value Description 1 An unmasked Timer B Time-Out interrupt has occurred. 0 A Timer B Time-Out interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `TBTOCINT` bit in the **GPTMICR** register.
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	TAMMIS	R/W	0	GPTM Timer A Mode Match Masked Interrupt Value Description 1 An unmasked Timer A Mode Match interrupt has occurred. 0 A Timer A Mode Match interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `TAMCINT` bit in the **GPTMICR** register.
3	RTCMIS	RO	0	GPTM RTC Masked Interrupt Value Description 1 An unmasked RTC event interrupt has occurred. 0 An RTC event interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `RTCCINT` bit in the **GPTMICR** register.
2	CAEMIS	RO	0	GPTM Capture A Event Masked Interrupt Value Description 1 An unmasked Capture A event interrupt has occurred. 0 A Capture A event interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `CAECINT` bit in the **GPTMICR** register.
1	CAMMIS	RO	0	GPTM Capture A Match Masked Interrupt Value Description 1 An unmasked Capture A Match interrupt has occurred. 0 A Capture A Mode Match interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `CAMCINT` bit in the **GPTMICR** register.

Bit/Field	Name	Type	Reset	Description
0	TATOMIS	RO	0	GPTM Timer A Time-Out Masked Interrupt Value Description 1 An unmasked Timer A Time-Out interrupt has occurred. 0 A Timer A Time-Out interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `TATOCINT` bit in the **GPTMICR** register.

Register 8: GPTM Interrupt Clear (GPTMICR), offset 0x024

This register is used to clear the status bits in the **GPTMRIS** and **GPTMMIS** registers. Writing a 1 to a bit clears the corresponding bit in the **GPTMRIS** and **GPTMMIS** registers.

GPTM Interrupt Clear (GPTMICR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x024
Type W1C, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				TBMCINT	CBECINT	CBMCINT	TBTOCINT	reserved			TAMCINT	RTCCINT	CAECINT	CAMCINT	TATOCINT
Type	RO	RO	RO	RO	W1C	W1C	W1C	W1C	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
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TBMCINT	W1C	0	GPTM Timer B Mode Match Interrupt Clear Writing a 1 to this bit clears the `TBMRIS` bit in the **GPTMRIS** register and the `TBMMIS` bit in the **GPTMMIS** register.
10	CBECINT	W1C	0	GPTM Capture B Event Interrupt Clear Writing a 1 to this bit clears the `CBERIS` bit in the **GPTMRIS** register and the `CBEMIS` bit in the **GPTMMIS** register.
9	CBMCINT	W1C	0	GPTM Capture B Match Interrupt Clear Writing a 1 to this bit clears the `CBMRIS` bit in the **GPTMRIS** register and the `CBMMIS` bit in the **GPTMMIS** register.
8	TBTOCINT	W1C	0	GPTM Timer B Time-Out Interrupt Clear Writing a 1 to this bit clears the `TBTORIS` bit in the **GPTMRIS** register and the `TBTOMIS` bit in the **GPTMMIS** register.
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	TAMCINT	W1C	0	GPTM Timer A Mode Match Interrupt Clear Writing a 1 to this bit clears the `TAMRIS` bit in the **GPTMRIS** register and the `TAMMIS` bit in the **GPTMMIS** register.
3	RTCCINT	W1C	0	GPTM RTC Interrupt Clear Writing a 1 to this bit clears the `RTCRIS` bit in the **GPTMRIS** register and the `RTCMIS` bit in the **GPTMMIS** register.

Bit/Field	Name	Type	Reset	Description
2	CAECINT	W1C	0	GPTM Capture A Event Interrupt Clear Writing a 1 to this bit clears the `CAERIS` bit in the **GPTMRIS** register and the `CAEMIS` bit in the **GPTMMIS** register.
1	CAMCINT	W1C	0	GPTM Capture A Match Interrupt Clear Writing a 1 to this bit clears the `CAMRIS` bit in the **GPTMRIS** register and the `CAMMIS` bit in the **GPTMMIS** register.
0	TATOCINT	W1C	0	GPTM Timer A Time-Out Raw Interrupt Writing a 1 to this bit clears the `TATORIS` bit in the **GPTMRIS** register and the `TATOMIS` bit in the **GPTMMIS** register.

Register 9: GPTM Timer A Interval Load (GPTMTAILR), offset 0x028

This register is used to load the starting count value into the timer. When GPTM is configured to one of the 32-bit modes, **GPTMTAILR** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM Timer B Interval Load (GPTMTBILR)** register). In 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of **GPTMTBILR**.

GPTM Timer A Interval Load (GPTMTAILR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x028
Type R/W, reset 0xFFFF.FFFF

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TAILRL															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:16	TAILRH	R/W	0xFFFF	GPTM Timer A Interval Load Register High When configured for 32-bit mode via the **GPTMCFG** register, the **GPTM Timer B 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 Timer A Interval Load Register Low For both 16- and 32-bit modes, writing this field loads the counter for Timer A. A read returns the current value of **GPTMTAILR**.

Register 10: GPTM Timer B Interval Load (GPTMTBILR), offset 0x02C

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

GPTM Timer B Interval Load (GPTMTBILR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x02C
Type R/W, reset 0x0000.FFFF

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TBILRL															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TBILRL	R/W	0xFFFF	GPTM Timer B Interval Load Register When the GPTM is not configured as a 32-bit timer, a write to this field updates **GPTMTBILR**. In 32-bit mode, writes are ignored, and reads return the current value of **GPTMTBILR**.

Register 11: GPTM Timer A Match (GPTMTAMATCHR), offset 0x030

This register is loaded with a match value. Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode. In 16-bit Edge-Count mode, this register along with **GPTMTAILR**, determines how many edge events are counted. The total number of edge events counted is equal to the value in **GPTMTAILR** minus this value.

GPTM Timer A Match (GPTMTAMATCHR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x030
Type R/W, reset 0xFFFF.FFFF

	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	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
	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	GPTM Timer A Match Register High When the timer is configured for 32-bit mode via the **GPTMCFG** register, this value is compared to the upper half of **GPTMTAR** to determine match events. In 16-bit mode, this field reads as 0 and does not have an effect on the state of **GPTMTBMATCHR**.
15:0	TAMRL	R/W	0xFFFF	GPTM Timer A Match Register Low When the timer is configured for 32-bit mode via the **GPTMCFG** register, this value is compared to the lower half of **GPTMTAR**, to determine match events. When the timer is configured for 16-bit mode via the **GPTMCFG** register, this value is compared to **GPTMTAR** to determine match events. When configured for 16-bit 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. When configured for PWM mode, this value along with **GPTMTAILR**, determines the duty cycle of the output PWM signal.

Register 12: GPTM Timer B Match (GPTMTBMATCHR), offset 0x034

This register is loaded with a match value. Interrupts can be generated when the timer value is equal to the value in this register in one-shot or periodic mode. In 16-bit Edge-Count mode, this register along with **GPTMTAILR**, determines how many edge events are counted. The total number of edge events counted is equal to the value in **GPTMTAILR** minus this value.

GPTM Timer B Match (GPTMTBMATCHR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x034
Type R/W, reset 0x0000.FFFF

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TBMRL															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TBMRL	R/W	0xFFFF	GPTM Timer B Match Register Low When the timer is configured for 16-bit mode via the **GPTMCFG** register, this value is compared to **GPTMTBR** to determine match events. When configured for 16-bit 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. When configured for PWM mode, this value along with **GPTMTBILR**, determines the duty cycle of the output PWM signal.

Register 13: GPTM Timer A Prescale (GPTMTAPR), offset 0x038

This register allows software to extend the range of the 16-bit timers.

GPTM Timer A Prescale (GPTMTAPR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TAPSR	R/W	0x00	GPTM Timer A Prescale The register loads this value on a write. A read returns the current value of the register. Refer to Table 11-3 on page 343 for more details and an example.

Register 14: GPTM Timer B Prescale (GPTMTBPR), offset 0x03C

This register allows software to extend the range of the 16-bit timers.

GPTM Timer B Prescale (GPTMTBPR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x03C
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								TBPSR							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TBPSR	R/W	0x00	GPTM Timer B Prescale The register loads this value on a write. A read returns the current value of this register. Refer to Table 11-3 on page 343 for more details and an example.

Register 15: GPTM Timer A (GPTMTAR), offset 0x048

This register shows the current value of the Timer A counter in all cases except for Input Edge-Count mode. When in this mode, this register contains the time at which the last edge event took place.

GPTM Timer A (GPTMTAR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x048
Type RO, reset 0xFFFF.FFFF

	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	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
	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	GPTM Timer A Register High If the **GPTMCFG** is in a 32-bit mode, Timer B value is read. If the **GPTMCFG** is in a 16-bit mode, this is read as zero.
15:0	TARL	RO	0xFFFF	GPTM Timer A Register Low A read returns the current value of the **GPTM Timer A Count Register**, except in Input Edge-Count mode, when it returns the timestamp from the last edge event.

Register 16: GPTM Timer B (GPTMTBR), offset 0x04C

This register shows the current value of the Timer B counter in all cases except for Input Edge-Count mode. When in this mode, this register contains the time at which the last edge event took place.

GPTM Timer B (GPTMTBR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x04C
Type RO, 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
	TBRL															
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	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TBRL	RO	0xFFFF	GPTM Timer B A read returns the current value of the **GPTM Timer B Count Register**, except in Input Edge-Count mode, when it returns the timestamp from the last edge event.

Register 17: GPTM Timer A Value (GPTMTAV), offset 0x050

This register shows the current, free-running value of Timer A in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry.

GPTM Timer A Value (GPTMTAV)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x050
Type RO, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TAVH															
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
	TAVL															
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	TAVH	RO	0xFFFF	GPTM Timer A Value High If the **GPTMCFG** is configured for 32-bit mode, the Timer B value is read. If the **GPTMCFG** is configured for 16-bit mode, this is read as zero.
15:0	TAVL	RO	0xFFFF	GPTM Timer A Register Low A read returns the current value of Timer A.

Register 18: GPTM Timer B Value (GPTMTBV), offset 0x054

This register shows the current, free-running value of Timer B in all modes. Software can use this value to determine the time elapsed between an interrupt and the ISR entry.

GPTM Timer B Value (GPTMTBV)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x054
Type RO, 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
	TBVL															
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	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TBVL	RO	0xFFFF	GPTM Timer B Register A read returns the current value of Timer B.

12 Watchdog Timers

A watchdog timer can generate a nonmaskable interrupt (NMI) 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 LM3S1W16 microcontroller has two Watchdog Timer Modules, one module is clocked by the system clock (Watchdog Timer 0) and the other is clocked by the PIOSC (Watchdog Timer 1). The two modules are identical except that WDT1 is in a different clock domain, and therefore requires synchronizers. As a result, WDT1 has a bit defined in the **Watchdog Timer Control (WDTCTL)** register to indicate when a write to a WDT1 register is complete. Software can use this bit to ensure that the previous access has completed before starting the next access.

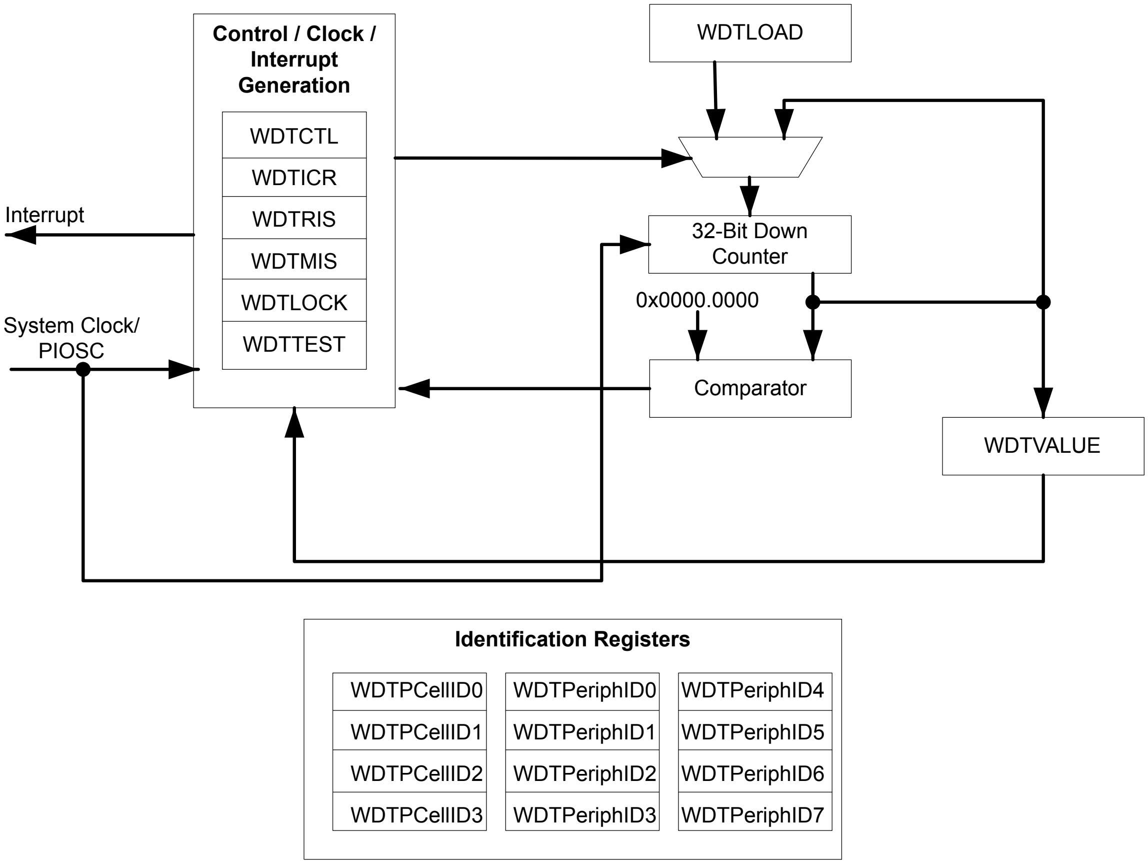
The Stellaris® LM3S1W16 controller has two Watchdog Timer modules with the following features:

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

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

12.1 Block Diagram

Figure 12-1. WDT Module Block Diagram



12.2 Functional Description

The Watchdog Timer module generates the first time-out signal when the 32-bit counter reaches the zero state after being enabled; enabling the counter also enables the watchdog timer interrupt. After the first time-out event, the 32-bit counter is re-loaded with the value of the **Watchdog Timer Load (WDTLOAD)** register, and the timer resumes counting down from that value. Once the Watchdog Timer has been configured, the **Watchdog Timer Lock (WDTLOCK)** register is written, which prevents the timer configuration from being inadvertently altered by software.

If the timer counts down to its zero state again before the first time-out interrupt is cleared, and the reset signal has been enabled by setting the `RESEN` bit in the **WDTCTL** register, the Watchdog timer asserts its reset signal to the system. If the interrupt is cleared before the 32-bit counter reaches its second time-out, the 32-bit counter is loaded with the value in the **WDTLOAD** register, and counting resumes from that value.

If **WDTLOAD** is written with a new value while the Watchdog Timer counter is counting, then the counter is loaded with the new value and continues counting.

Writing to **WDTLOAD** does not clear an active interrupt. An interrupt must be specifically cleared by writing to the **Watchdog Interrupt Clear (WDTICR)** register.

The Watchdog module interrupt and reset generation can be enabled or disabled as required. When the interrupt is re-enabled, the 32-bit counter is preloaded with the load register value and not its last state.

12.2.1 Register Access Timing

Because the Watchdog Timer 1 module has an independent clocking domain, its registers must be written with a timing gap between accesses. Software must guarantee that this delay is inserted between back-to-back writes to WDT1 registers or between a write followed by a read to the registers. The timing for back-to-back reads from the WDT1 module has no restrictions. The `WRC` bit in the **Watchdog Control (WDTCTL)** register for WDT1 indicates that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **WDTCTL** for `WRC`=1 prior to accessing another register. Note that WDT0 does not have this restriction as it runs off the system clock.

12.3 Initialization and Configuration

To use the WDT, its peripheral clock must be enabled by setting the `WDT` bit in the **RCGC0** register, see page 141.

The Watchdog Timer is configured using the following sequence:

1. Load the **WDTLOAD** register with the desired timer load value.
2. If WDT1, wait for the `WRC` bit in the **WDTCTL** register to be set.
3. If the Watchdog is configured to trigger system resets, set the `RESEN` bit in the **WDTCTL** register.
4. If WDT1, wait for the `WRC` bit in the **WDTCTL** register to be set.
5. Set the `INTEN` bit in the **WDTCTL** register to enable the Watchdog and lock the control register.

If software requires that all of the watchdog registers are locked, the Watchdog Timer module can be fully locked by writing any value to the **WDTLOCK** register. To unlock the Watchdog Timer, write a value of 0x1ACC.E551.

12.4 Register Map

Table 12-1 on page 384 lists the Watchdog registers. The offset listed is a hexadecimal increment to the register’s address, relative to the Watchdog Timer base address:

- WDT0: 0x4000.0000
- WDT1: 0x4000.1000

Note that the Watchdog Timer module clock must be enabled before the registers can be programmed (see page 141).

Table 12-1. Watchdog Timers Register Map

Offset	Name	Type	Reset	Description	See page
0x000	WDTLOAD	R/W	0xFFFF.FFFF	Watchdog Load	385
0x004	WDTVALUE	RO	0xFFFF.FFFF	Watchdog Value	386
0x008	WDTCTL	R/W	0x0000.0000 (WDT0) 0x8000.0000 (WDT1)	Watchdog Control	387
0x00C	WDTICR	WO	-	Watchdog Interrupt Clear	389
0x010	WDTRIS	RO	0x0000.0000	Watchdog Raw Interrupt Status	390
0x014	WDTMIS	RO	0x0000.0000	Watchdog Masked Interrupt Status	391
0x418	WDTTEST	R/W	0x0000.0000	Watchdog Test	392
0xC00	WDTLOCK	R/W	0x0000.0000	Watchdog Lock	393
0xFD0	WDTPeriphID4	RO	0x0000.0000	Watchdog Peripheral Identification 4	394
0xFD4	WDTPeriphID5	RO	0x0000.0000	Watchdog Peripheral Identification 5	395
0xFD8	WDTPeriphID6	RO	0x0000.0000	Watchdog Peripheral Identification 6	396
0xFDC	WDTPeriphID7	RO	0x0000.0000	Watchdog Peripheral Identification 7	397
0xFE0	WDTPeriphID0	RO	0x0000.0005	Watchdog Peripheral Identification 0	398
0xFE4	WDTPeriphID1	RO	0x0000.0018	Watchdog Peripheral Identification 1	399
0xFE8	WDTPeriphID2	RO	0x0000.0018	Watchdog Peripheral Identification 2	400
0xFEC	WDTPeriphID3	RO	0x0000.0001	Watchdog Peripheral Identification 3	401
0xFF0	WDTPCellID0	RO	0x0000.000D	Watchdog PrimeCell Identification 0	402
0xFF4	WDTPCellID1	RO	0x0000.00F0	Watchdog PrimeCell Identification 1	403
0xFF8	WDTPCellID2	RO	0x0000.0006	Watchdog PrimeCell Identification 2	404
0xFFC	WDTPCellID3	RO	0x0000.00B1	Watchdog PrimeCell Identification 3	405

12.5 Register Descriptions

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

Register 1: Watchdog Load (WDTLOAD), offset 0x000

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

Watchdog Load (WDTLOAD)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x000
Type R/W, reset 0xFFFF.FFFF

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WDTLOAD															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:0	WDTLOAD	R/W	0xFFFF.FFFF	Watchdog Load Value

Register 2: Watchdog Value (WDTVALUE), offset 0x004

This register contains the current count value of the timer.

Watchdog Value (WDTVALUE)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x004
Type RO, reset 0xFFFF.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WDTVALUE															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WDTVALUE															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:0	WDTVALUE	RO	0xFFFF.FFFF	Watchdog Value Current value of the 32-bit down counter.

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

This register is the watchdog control register. The watchdog timer can be configured to generate a reset signal (on second time-out) or an interrupt on time-out.

When the watchdog interrupt has been enabled, all subsequent writes to the control register are ignored. The only mechanism that can re-enable writes is a hardware reset.

Important: Because the Watchdog Timer 1 module has an independent clocking domain, its registers must be written with a timing gap between accesses. Software must guarantee that this delay is inserted between back-to-back writes to WDT1 registers or between a write followed by a read to the registers. The timing for back-to-back reads from the WDT1 module has no restrictions. The `WRC` bit in the **Watchdog Control (WDTCTL)** register for WDT1 indicates that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **WDTCTL** for `WRC`=1 prior to accessing another register. Note that WDT0 does not have this restriction as it runs off the system clock and therefore does not have a `WRC` bit.

Watchdog Control (WDTCTL)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x008
Type R/W, reset 0x0000.0000 (WDT0) and 0x8000.0000 (WDT1)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WRC	reserved														
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														RESEN	INTEN
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	WRC	RO	1	Write Complete The `WRC` values are defined as follows: Value Description 0 A write access to one of the WDT1 registers is in progress. 1 A write access is not in progress, and WDT1 registers can be read or written. **Note:** This bit is reserved for WDT0 and has a reset value of 0.
30:2	reserved	RO	0x000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
1	RESEN	R/W	0	Watchdog Reset Enable The `RESEN` values are defined as follows: Value Description 0 Disabled. 1 Enable the Watchdog module reset output.
0	INTEN	R/W	0	Watchdog Interrupt Enable The `INTEN` values are defined as follows: Value Description 0 Interrupt event disabled (once this bit is set, it can only be cleared by a hardware reset). 1 Interrupt event enabled. Once enabled, all writes are ignored.

Register 4: Watchdog Interrupt Clear (WDTICR), offset 0x00C

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

Watchdog Interrupt Clear (WDTICR)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x00C
Type WO, reset -

	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)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x010
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															WDTRIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Value	Description
1	A watchdog time-out event has occurred.
0	The watchdog has not timed out.

Register 6: Watchdog Masked Interrupt Status (WDTMIS), offset 0x014

This register is the masked interrupt status register. The value of this register is the logical AND of the raw interrupt bit and the Watchdog interrupt enable bit.

Watchdog Masked Interrupt Status (WDTMIS)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x014
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WDTMIS	RO	0	Watchdog Masked Interrupt Status

Value	Description
1	A watchdog time-out event has been signalled to the interrupt controller.
0	The watchdog has not timed out or the watchdog timer interrupt is masked.

Register 7: Watchdog Test (WDTTEST), offset 0x418

This register provides user-enabled stalling when the microcontroller asserts the CPU halt flag during debug.

Watchdog Test (WDTTEST)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0x418
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	STALL	R/W	0	Watchdog Stall Enable Value Description 1 If the microcontroller is stopped with a debugger, the watchdog timer stops counting. Once the microcontroller is restarted, the watchdog timer resumes counting. 0 The watchdog timer continues counting if the microcontroller is stopped with a debugger.
7:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 8: Watchdog Lock (WDTLOCK), offset 0xC00

Writing 0x1ACC.E551 to the **WDTLOCK** register enables write access to all other registers. Writing any other value to the **WDTLOCK** register re-enables the locked state for register writes to all the other registers. Reading the **WDTLOCK** register returns the lock status rather than the 32-bit value written. Therefore, when write accesses are disabled, reading the **WDTLOCK** register returns 0x0000.0001 (when locked; otherwise, the returned value is 0x0000.0000 (unlocked)).

Watchdog Lock (WDTLOCK)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xC00
Type R/W, reset 0x0000.0000

	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	0x0000.0000	Watchdog Lock A write of the value 0x1ACC.E551 unlocks the watchdog registers for write access. A write of any other value reapplies the lock, preventing any register updates. A read of this register returns the following values:

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

Register 9: Watchdog Peripheral Identification 4 (WDTPeriphID4), offset 0xFD0

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

Watchdog Peripheral Identification 4 (WDTPeriphID4)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFD0
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	WDT Peripheral ID Register [7:0]

Register 10: Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4

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

Watchdog Peripheral Identification 5 (WDTPeriphID5)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFD4
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	WDT Peripheral ID Register [15:8]

Register 11: Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8

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

Watchdog Peripheral Identification 6 (WDTPeriphID6)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFD8
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID6							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	WDT Peripheral ID Register [23:16]

Register 12: Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC

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

Watchdog Peripheral Identification 7 (WDTPeriphID7)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFDC
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID7							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	WDT Peripheral ID Register [31:24]

Register 13: Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0

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

Watchdog Peripheral Identification 0 (WDTPeriphID0)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFE0
Type RO, reset 0x0000.0005

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x05	Watchdog Peripheral ID Register [7:0]

Register 14: Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4

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

Watchdog Peripheral Identification 1 (WDTPeriphID1)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFE4
Type RO, reset 0x0000.0018

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x18	Watchdog Peripheral ID Register [15:8]

Register 15: Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8

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

Watchdog Peripheral Identification 2 (WDTPeriphID2)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFE8
Type RO, reset 0x0000.0018

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	Watchdog Peripheral ID Register [23:16]

Register 16: Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC

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

Watchdog Peripheral Identification 3 (WDTPeriphID3)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFEC
Type RO, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	Watchdog Peripheral ID Register [31:24]

Register 17: Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0

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

Watchdog PrimeCell Identification 0 (WDTPCellID0)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFF0
Type RO, reset 0x0000.000D

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	Watchdog PrimeCell ID Register [7:0]

Register 18: Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4

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

Watchdog PrimeCell Identification 1 (WDTPCellID1)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFF4
Type RO, reset 0x0000.00F0

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	Watchdog PrimeCell ID Register [15:8]

Register 19: Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8

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

Watchdog PrimeCell Identification 2 (WDTPCellID2)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFF8
Type RO, reset 0x0000.0006

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x06	Watchdog PrimeCell ID Register [23:16]

Register 20: Watchdog PrimeCell Identification 3 (WDTPCellID3), offset 0xFFC

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

Watchdog PrimeCell Identification 3 (WDTPCellID3)

WDT0 base: 0x4000.0000
WDT1 base: 0x4000.1000
Offset 0xFFC
Type RO, reset 0x0000.00B1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	Watchdog PrimeCell ID Register [31:24]

13 Analog-to-Digital Converter (ADC)

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

The Stellaris® ADC module features 10-bit conversion resolution and supports eight input channels, plus an internal temperature sensor. The ADC module contains four programmable sequencers allowing the sampling of multiple analog input sources without controller intervention. Each sample sequencer provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequencer priority. A digital comparator function is included which allows the conversion value to be diverted to a digital comparator module. The digital comparator module provides eight digital comparator. The comparator module measures the ADC conversion value against two user-defined values to determine the operational range of the signal.

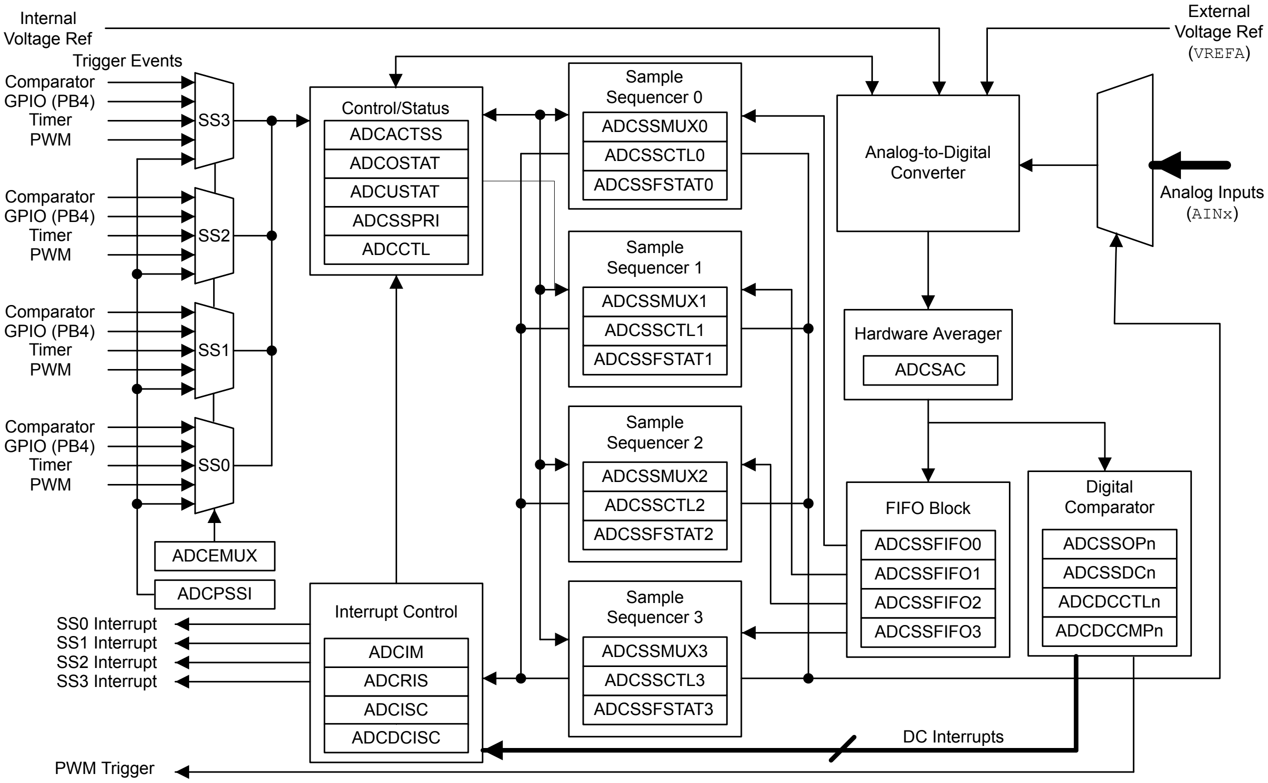
The Stellaris® LM3S1W16 microcontroller provides one ADC module with the following features:

- Eight analog input channels
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Sample rate of one million samples/second
- Flexible, configurable analog-to-digital conversion
- Four programmable sample conversion sequencers from one to eight entries long, with corresponding conversion result FIFOs
- Flexible trigger control
  - Controller (software)
  - Timers
  - Analog Comparators
  - GPIO
- Hardware averaging of up to 64 samples for improved accuracy
- Digital comparison unit providing eight digital comparators
- Converter uses an internal 3-V reference or an external reference
- Power and ground for the analog circuitry is separate from the digital power and ground
- Efficient transfers using Micro Direct Memory Access Controller (µDMA)
  - Dedicated channel for each sample sequencer
  - Burst request asserted when interrupt is triggered

13.1 Block Diagram

Figure 13-1 on page 407 provides details on the internal configuration of the ADC controls and data registers.

Figure 13-1. ADC Module Block Diagram



13.2 Signal Description

Table 13-1 on page 407 lists the external signals of the ADC module and describes the function of each. The ADC signals are analog functions for some GPIO signals. The column in the table below titled "Pin Mux/Pin Assignment" lists the GPIO pin placement for the ADC signals. Note that when a pin is used as an ADC input, the appropriate bit in the **GPIO Analog Mode Select (GPIOAMSEL)** register must be set to disable the analog isolation circuit, and the appropriate bit in the **GPIO Digital Enable (GPIODEN)** register must be clear to disable digital function. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 285.

Table 13-1. Signals for ADC

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`AIN0`	1	PE3	I	Analog	Analog-to-digital converter input 0.
`AIN1`	2	PE2	I	Analog	Analog-to-digital converter input 1.
`AIN2`	5	PE1	I	Analog	Analog-to-digital converter input 2.
`AIN3`	6	PE0	I	Analog	Analog-to-digital converter input 3.
`AIN4`	64	PD3	I	Analog	Analog-to-digital converter input 4.
`AIN5`	63	PD2	I	Analog	Analog-to-digital converter input 5.

Table 13-1. Signals for ADC *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`AIN6`	62	PD1	I	Analog	Analog-to-digital converter input 6.
`AIN7`	61	PD0	I	Analog	Analog-to-digital converter input 7.
`VREFA`	56	PB6	I	Analog	This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to `VREFA` is the voltage with which an `AIN`n signal is converted to 1023. The `VREFA` input is limited to the range specified in Table 21-2 on page 643.

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

13.3 Functional Description

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

13.3.1 Sample Sequencers

The sampling control and data capture is handled by the sample sequencers. All of the sequencers are identical in implementation except for the number of samples that can be captured and the depth of the FIFO. Table 13-2 on page 408 shows the maximum number of samples that each sequencer can capture and its corresponding FIFO depth. In this implementation, each FIFO entry is a 32-bit word, with the lower 10 bits containing the conversion result.

Table 13-2. Samples and FIFO Depth of Sequencers

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

For a given sample sequence, each sample is defined by two 4-bit nibbles in the **ADC Sample Sequence Input Multiplexer Select (ADCSSMUXn)** and **ADC Sample Sequence Control (ADCSSCTLn)** registers, where "n" corresponds to the sequence number. The **ADCSSMUXn** nibbles select the input pin, while the **ADCSSCTLn** nibbles contain the sample control bits corresponding to parameters such as temperature sensor selection, interrupt enable, end of sequence, and differential input mode. Sample sequencers are enabled by setting the respective ASENn bit in the **ADC Active Sample Sequencer (ADCACTSS)** register and should be configured before being enabled. Sampling is then initiated by setting the SSn bit in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register.

When configuring a sample sequence, multiple uses of the same input pin within the same sequence is allowed. In the **ADCSSCTLn** register, the `IE`n 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 (ADCSSFIFOn)** registers. The FIFOs are simple circular buffers that read a single address to "pop" result data. For software debug purposes, the positions of the FIFO head and tail pointers are visible in the **ADC Sample Sequence FIFO Status (ADCSSFSTATn)** registers along with `FULL` and `EMPTY` status flags. Overflow and underflow conditions are monitored using the **ADCOSTAT** and **ADCUSTAT** registers.

13.3.2 Module Control

Outside of the sample sequencers, the remainder of the control logic is responsible for tasks such as:

- Interrupt generation
- Sequence prioritization
- Trigger configuration
- Comparator configuration

Most of the ADC control logic runs at the ADC clock rate of 14-18 MHz. The internal ADC divider is configured for 16-MHz operation automatically by hardware when the system `XTAL` is selected.

13.3.2.1 Interrupts

The register configurations of the sample sequencers and digital comparators dictate which events generate raw interrupts, but do not have control over whether the interrupt is actually sent to the interrupt controller. The ADC module's interrupt signals are controlled by the state of the `MASK` bits in the **ADC Interrupt Mask (ADCIM)** register. Interrupt status can be viewed at two locations: the **ADC Raw Interrupt Status (ADCRIS)** register, which shows the raw status of the various interrupt signals; and the **ADC Interrupt Status and Clear (ADCISC)** register, which shows active interrupts that are enabled by the **ADCIM** register. Sequencer interrupts are cleared by writing a 1 to the corresponding `IN` bit in **ADCISC**. Digital comparator interrupts are cleared by writing a 1 to the **ADC Digital Comparator Interrupt Status and Clear (ADCDCISC)** register.

13.3.2.2 DMA Operation

The ADC module provides a request signal to the μDMA controller for each sample sequencer. Each sample sequencer has a dedicated μDMA channel. The request signal is a burst type and is asserted whenever an interrupt is enabled in a sample sequence (`IE` bit in the **ADCSSCTLn** register is set). Single requests are not supported.

The arbitration size of the μDMA transfer must be a power of 2, and the associated `IE` bits in the **ADDSSCTLn** register must be set. For example, if the μDMA channel of SS0 has an arbitration size of four, the `IE3` bit (4th sample) and the `IE7` bit (8th sample) must be set. Thus the μDMA request occurs every time 4 samples have been acquired. No other special steps are needed to enable the ADC module for μDMA operation.

Refer to the “Micro Direct Memory Access (μDMA)” on page 227 for more details about programming the μDMA controller.

13.3.2.3 Prioritization

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

13.3.2.4 Sampling Events

Sample triggering for each sample sequencer is defined in the **ADC Event Multiplexer Select (ADCEMUX)** register. Trigger sources include processor (default), analog comparators, an external signal on GPIO PB4, a GP Timer, and continuous sampling. Software can initiate sampling by setting the SSx bits in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register.

Care must be taken when using the continuous sampling trigger. If a sequencer's priority is too high, it is possible to starve other lower priority sequencers.

13.3.2.5 External Voltage Reference

An external reference voltage may be provided to serve as the maximum conversion value reference. The VREF bit in th**e ADC Control (ADCCTL)** register specifies whether to use the internal or external reference. The V_{REFA} specification defines the useful range for the external voltage reference, see Table 21-2 on page 643. Ground is always used as the reference level for the minimum conversion value. Care must be taken to supply a reference voltage of acceptable quality.

13.3.3 Hardware Sample Averaging Circuit

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

By default the averaging circuit is off, and all data from the converter passes through to the sequencer FIFO. The averaging hardware is controlled by the **ADC Sample Averaging Control (ADCSAC)** register (see page 442). A single averaging circuit has been implemented, thus all input channels receive the same amount of averaging whether they are single-ended or differential.

13.3.4 Analog-to-Digital Converter

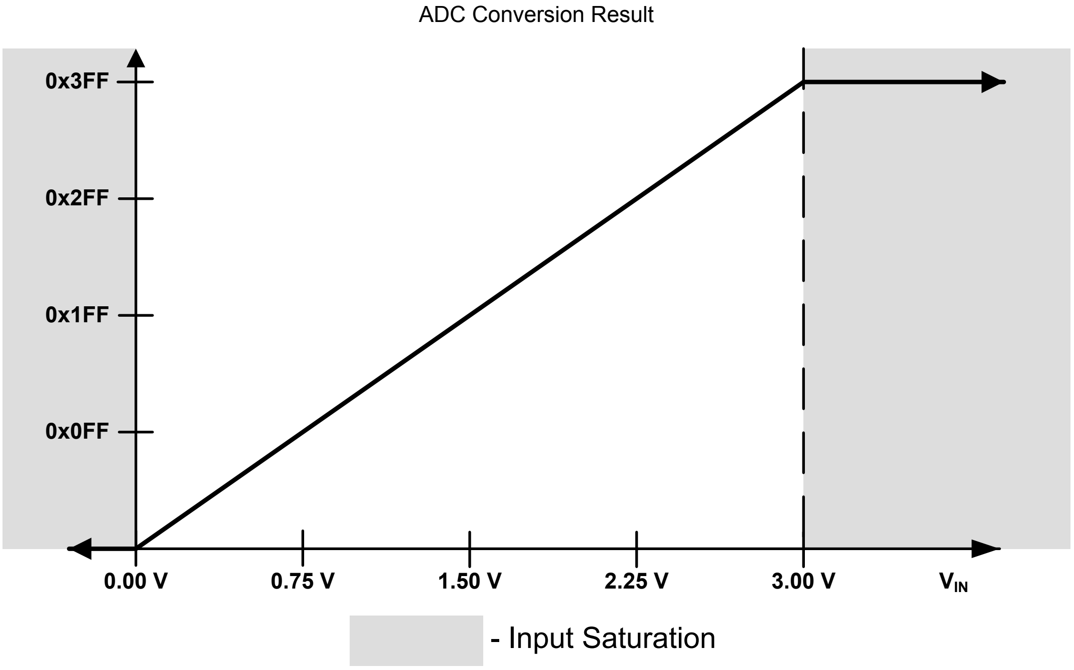
The Analog-to-Digital Converter (ADC) module uses a Successive Approximation Register (SAR) architecture to deliver a 10-bit, low-power, high-precision conversion value. The successive-approximation algorithm uses a current mode D/A converter to achieve lower settling time, resulting in higher conversion speeds for the A/D converter. In addition, built-in sample-and-hold circuitry with offset-calibration circuitry improves conversion accuracy. The ADC must be run from the PLL or a 14- to 18-MHz clock source.

The ADC operates from the 3.3-V analog and 1.2-V digital power supply. Integrated shutdown modes are available to reduce power consumption when ADC conversions are not required. The analog inputs are connected to the ADC through custom pads and specially balanced input paths to minimize the distortion on the inputs. Detailed information on the ADC power supplies and analog inputs can be found in “Analog-to-Digital Converter” on page 653.

13.3.4.1 Internal Voltage Reference

The band-gap circuitry generates an internal 3.0 V reference that can be used by the ADC to produce a conversion value from the selected analog input. The range of this conversion value is from 0x000 to 0x3FF. In single-ended-input mode, the 0x000 value corresponds to an analog input voltage of 0.0 V; the 0x3FF value corresponds to an analog input voltage of 3.0 V. This configuration results in a resolution of approximately 2.9 mV per ADC code. While the analog input pads can handle voltages beyond this range, the ADC conversions saturate in under-voltage and over-voltage cases. Figure 13-2 on page 411 shows the ADC conversion function of the analog inputs.

Figure 13-2. Internal Voltage Conversion Result

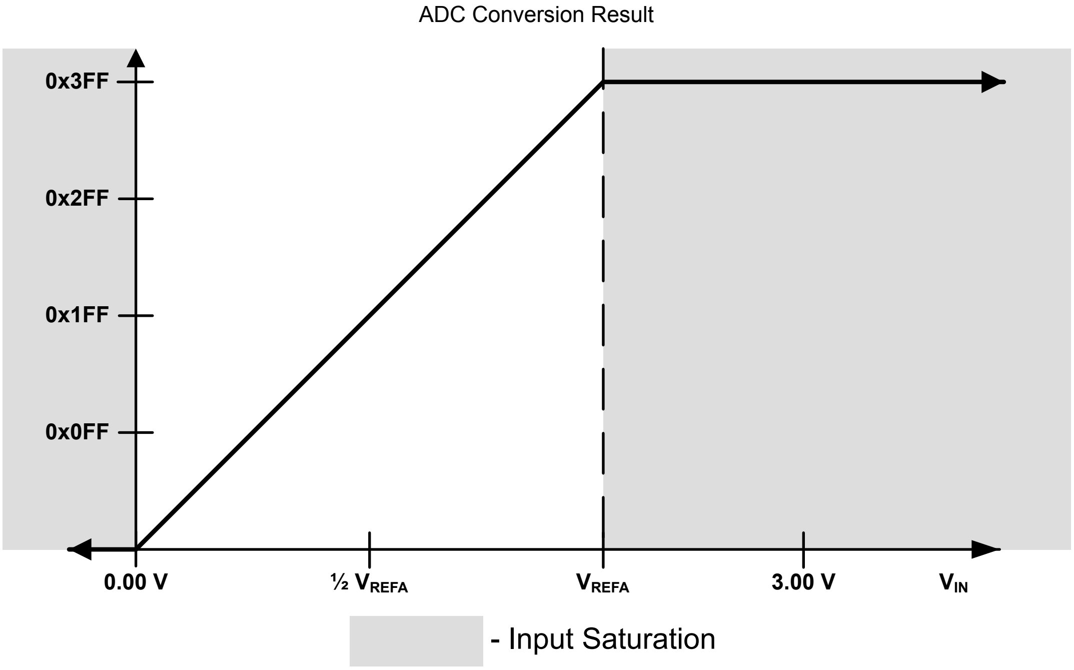


13.3.4.2 External Voltage Reference

The ADC can use an external voltage reference to produce the conversion value from the selected analog input by setting the VREF bit in the **ADC Control (ADCCTL)** register. While the range of the conversion value remains the same (0x000 to 0x3FF), the analog voltage associated with the 0x3FF value corresponds to the value of the external voltage reference, resulting in a smaller voltage resolution per ADC code. Analog input voltages above the external voltage reference saturate to 0x3FF while those below 0.0 V continue to saturate at 0x000. Figure 13-3 on page 412 shows the ADC conversion function of the analog inputs when using an external voltage reference.

The external voltage reference can be more accurate than the internal reference by using a high-precision source or trimming the source.

Figure 13-3. External Voltage Conversion Result



13.3.5 Differential Sampling

In addition to traditional single-ended sampling, the ADC module supports differential sampling of two analog input channels. To enable differential sampling, software must set the `Dn` bit in the **ADCSSCTL0n** register in a step's configuration nibble.

When a sequence step is configured for differential sampling, the input pair to sample must be configured in the **ADCSSMUXn** register. Differential pair 0 samples analog inputs 0 and 1; differential pair 1 samples analog inputs 2 and 3; and so on (see Table 13-3 on page 412). The ADC does not support other differential pairings such as analog input 0 with analog input 3.

Table 13-3. Differential Sampling Pairs

Differential Pair	Analog Inputs
0	0 and 1
1	2 and 3
2	4 and 5
3	6 and 7

The voltage sampled in differential mode is the difference between the odd and even channels:

ΔV (differential voltage) = V_{IN_EVEN} (even channels) – V_{IN_ODD} (odd channels), therefore:

- If $\Delta V = 0$, then the conversion result = 0x1FF
- If $\Delta V > 0$, then the conversion result > 0x1FF (range is 0x1FF–0x3FF)
- If $\Delta V < 0$, then the conversion result < 0x1FF (range is 0–0x1FF)

The differential pairs assign polarities to the analog inputs: the even-numbered input is always positive, and the odd-numbered input is always negative. In order for a valid conversion result to appear, the negative input must be in the range of ± 1.5 V of the positive input. If an analog input is greater than 3 V or less than 0 V (the valid range for analog inputs), the input voltage is clipped, meaning it appears as either 3 V or 0 V, respectively, to the ADC.

Figure 13-4 on page 413 shows an example of the negative input centered at 1.5 V. In this configuration, the differential range spans from -1.5 V to 1.5 V. Figure 13-5 on page 414 shows an example where the negative input is centered at -0.75 V, meaning inputs on the positive input saturate past a differential voltage of -0.75 V since the input voltage is less than 0 V. Figure 13-6 on page 414 shows an example of the negative input centered at 2.25 V, where inputs on the positive channel saturate past a differential voltage of 0.75 V since the input voltage would be greater than 3 V.

Figure 13-4. Differential Sampling Range, V_{IN_ODD} = 1.5 V

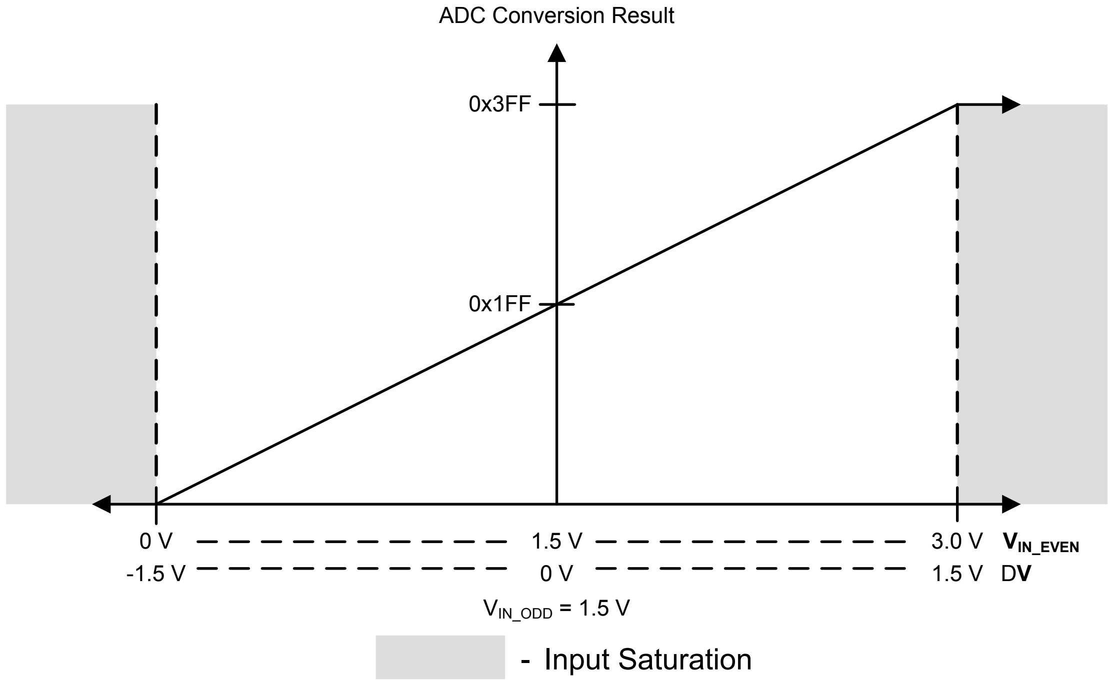


Figure 13-5. Differential Sampling Range, V_{IN_ODD} = 0.75 V

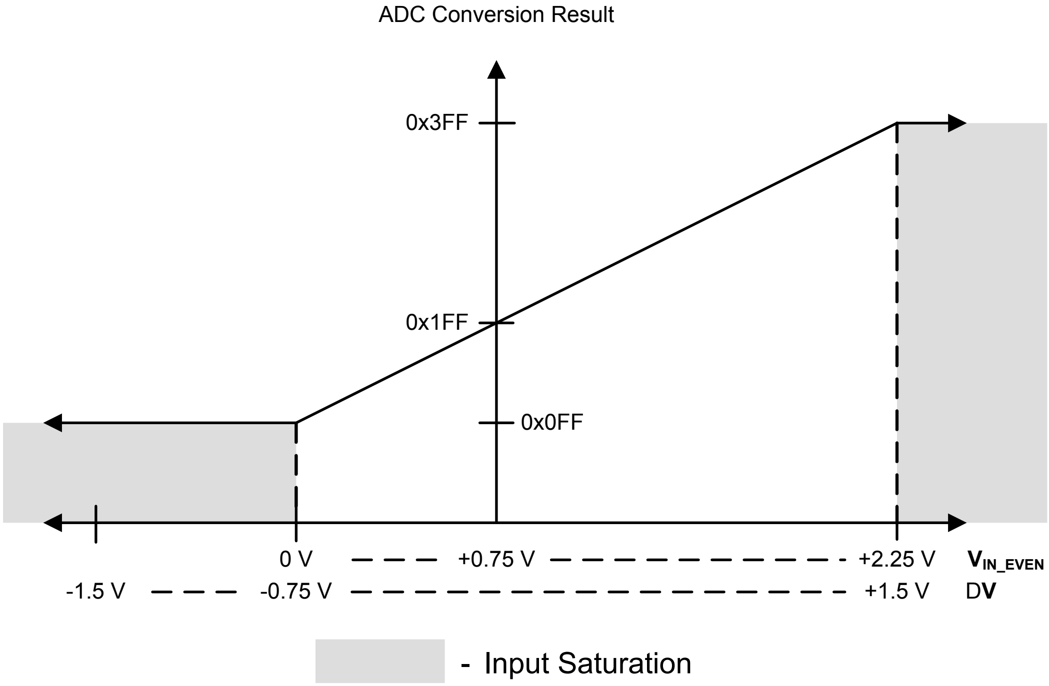
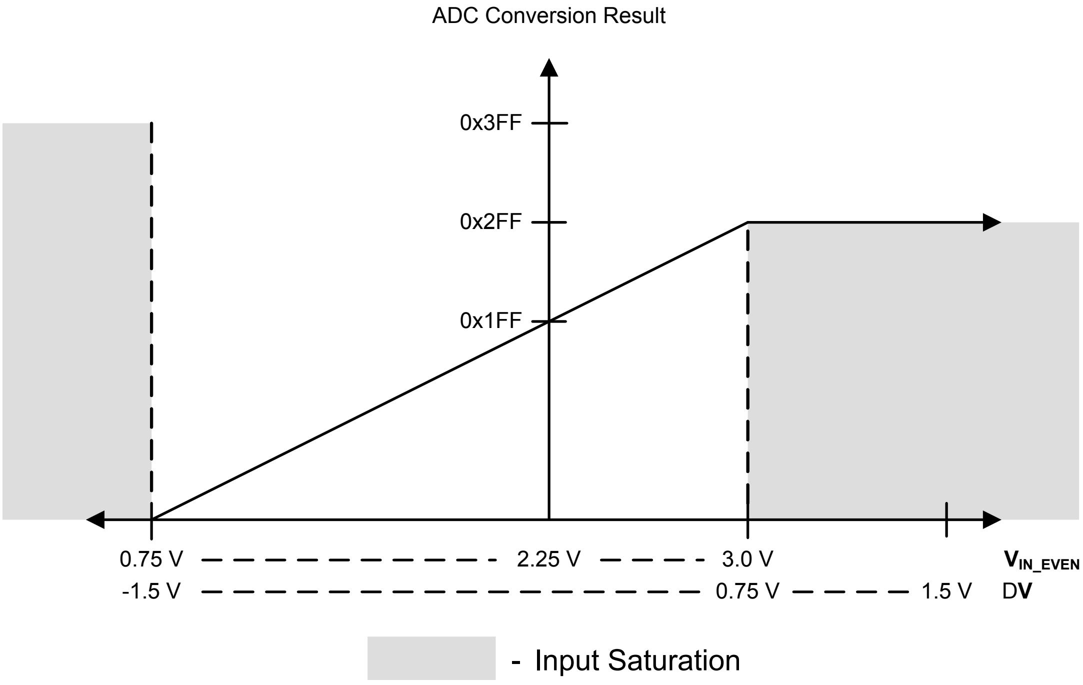


Figure 13-6. Differential Sampling Range, V_{IN_ODD} = 2.25 V



13.3.6 Internal Temperature Sensor

The temperature sensor serves two primary purposes: 1) to notify the system that internal temperature is too high or low for reliable operation and 2) to provide temperature measurements for calibration of the Hibernate module RTC trim value.

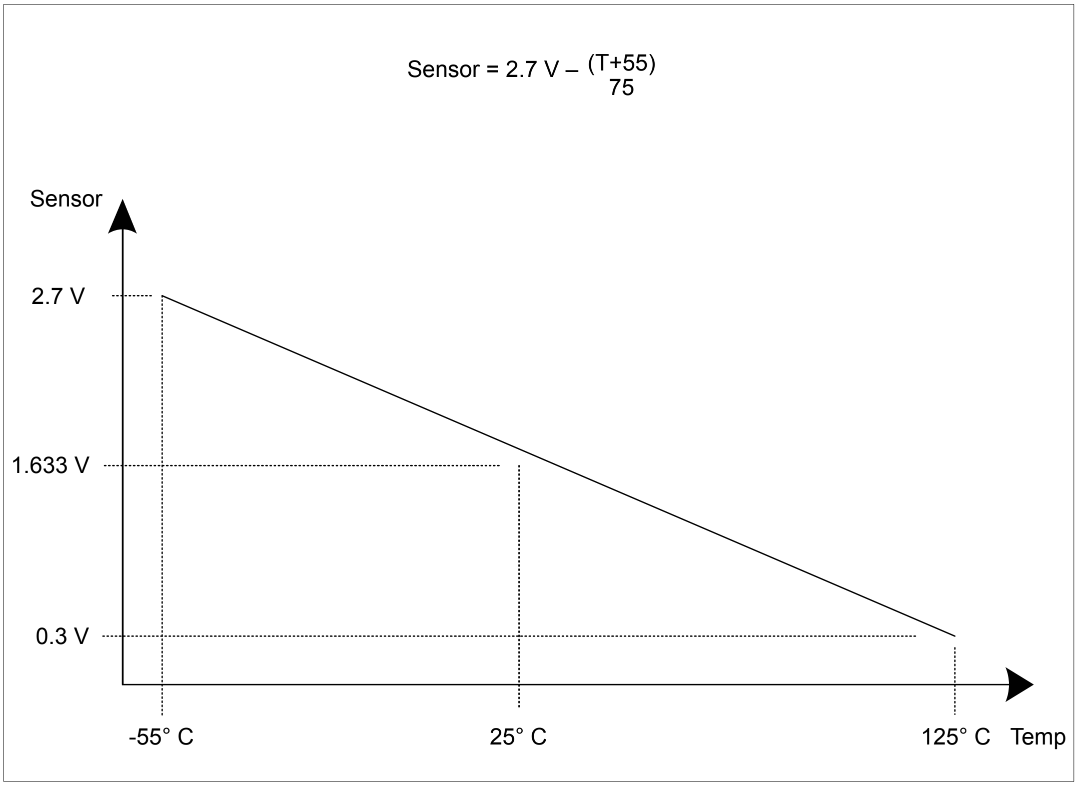
The temperature sensor does not have a separate enable, because it also contains the bandgap reference and must always be enabled. The reference is supplied to other analog modules; not just the ADC. In addition, the temperature sensor has a second power-down input in the 3.3 V domain which provides control by the Hibernation module.

The internal temperature sensor provides an analog temperature reading as well as a reference voltage. The voltage at the output terminal *SENSO* is given by the following equation:

`SENSO = 2.7 - ((T + 55) / 75)`

This relation is shown in Figure 13-7 on page 415.

Figure 13-7. Internal Temperature Sensor Characteristic



The temperature reading from the temperature sensor can also be given as a function of the ADC value. The following formula calculates temperature (in ℃) based on the ADC reading:

`Temperature = 147.5 - ((225 × ADC) / 1023)`

13.3.7 Digital Comparator Unit

An ADC is commonly used to sample an external signal and to monitor its value to ensure that it remains in a given range. To automate this monitoring procedure and reduce the amount of processor overhead that is required, eight digital comparator are provided. Conversions from the ADC that are sent to the digital comparators are compared against the user programmable limits in the **ADC**

Digital Comparator Range (ADCDCCMPn) registers. If the observed signal moves out of the acceptable range, a processor interrupt can be generated. The digital comparators four operational modes (Once, Always, Hysteresis Once, Hysteresis Always) can be applied to three separate regions (low band, mid band, high band) as defined by the user.

13.3.7.1 Output Functions

ADC conversions can either be stored in the ADC Sample Sequence FIFOs or compared using the digital comparator resources as defined by the `SnDCOP` bits in the **ADC Sample Sequence n Operation (ADCSSOPn)** register. These selected ADC conversions are used by their respective digital comparator to monitor the external signal. Each comparator has two possible output functions: processor interrupts and triggers.

Each function has its own state machine to track the monitored signal. Even though the interrupt and trigger functions can be enabled individually or both at the same time, the same conversion data is used by each function to determine if the right conditions have been met to assert the associated output.

Interrupts

The digital comparator interrupt function is enabled by setting the `CIE` bit in the **ADC Digital Comparator Control (ADCDCCTLn)** register. This bit enables the interrupt function state machine to start monitoring the incoming ADC conversions. When the appropriate set of conditions is met, and the `DCONSSx` bit is set in the **ADCIM** register, an interrupt is sent to the interrupt controller.

Triggers

The digital comparator trigger function is enabled by setting the `CTE` bit in the **ADCDCCTLn** register. This bit enables the trigger function state machine to start monitoring the incoming ADC conversions. When the appropriate set of conditions is met, the corresponding digital comparator trigger to the PWM module is asserted

13.3.7.2 Operational Modes

Four operational modes are provided to support a broad range of applications and multiple possible signaling requirements: Always, Once, Hysteresis Always, and Hysteresis Once. The operational mode is selected using the `CIM` or `CTM` field in the **ADCDCCTLn** register.

Always Mode

In the Always operational mode, the associated interrupt or trigger is asserted whenever the ADC conversion value meets its comparison criteria. The result is a string of assertions on the interrupt or trigger while the conversions are within the appropriate range.

Once Mode

In the Once operational mode, the associated interrupt or trigger is asserted whenever the ADC conversion value meets its comparison criteria, and the previous ADC conversion value did not. The result is a single assertion of the interrupt or trigger when the conversions are within the appropriate range.

Hysteresis-Always Mode

The Hysteresis-Always operational mode can only be used in conjunction with the low-band or high-band regions because the mid-band region must be crossed and the opposite region entered to clear the hysteresis condition. In the Hysteresis-Always mode, the associated interrupt or trigger is asserted in the following cases: 1) the ADC conversion value meets its comparison criteria or 2) a previous ADC conversion value has met the comparison criteria, and the hysteresis condition has not been cleared by entering the opposite region. The result is a string of assertions on the interrupt or trigger that continue until the opposite region is entered.

Hysteresis-Once Mode

The Hysteresis-Once operational mode can only be used in conjunction with the low-band or high-band regions because the mid-band region must be crossed and the opposite region entered to clear the hysteresis condition. In the Hysteresis-Once mode, the associated interrupt or trigger is asserted only when the ADC conversion value meets its comparison criteria, the hysteresis condition is clear, and the previous ADC conversion did not meet the comparison criteria. The result is a single assertion on the interrupt or trigger.

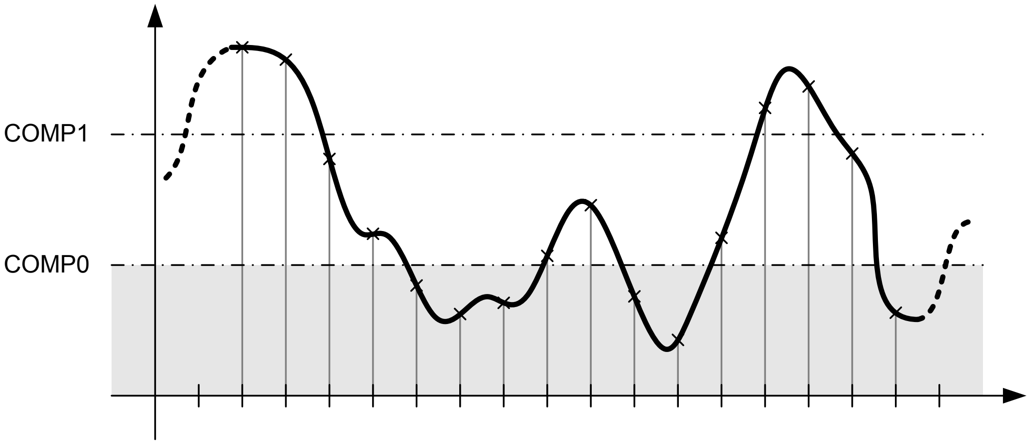
13.3.7.3 Function Ranges

The two comparison values, `COMP0` and `COMP1`, in the **ADC Digital Comparator Range (ADCDCCMPn)** register effectively break the conversion area into three distinct regions. These regions are referred to as the low-band (less than or equal to `COMP0`), mid-band (greater than `COMP0` but less than or equal to `COMP1`), and high-band (greater than `COMP1`) regions. `COMP0` and `COMP1` may be programmed to the same value, effectively creating two regions, but `COMP1` must always be greater than or equal to the value of `COMP0`. A `COMP1` value that is less than `COMP0` generates unpredictable results.

Low-Band Operation

To operate in the low-band region, either the `CIC` field or the `CTC` field in the **ADCDCCTLn** register must be programmed to 0x0. This setting causes interrupts or triggers to be generated in the low-band region as defined by the programmed operational mode. An example of the state of the interrupt/trigger signal in the low-band region for each of the operational modes is shown in Figure 13-8 on page 418. Note that a "0" in a column following the operational mode name (Always, Once, Hysteresis Always, and Hysteresis Once) indicates that the interrupt or trigger signal is de-asserted and a "1" indicates that the signal is asserted.

Figure 13-8. Low-Band Operation (CIC=0x0 and/or CTC=0x0)

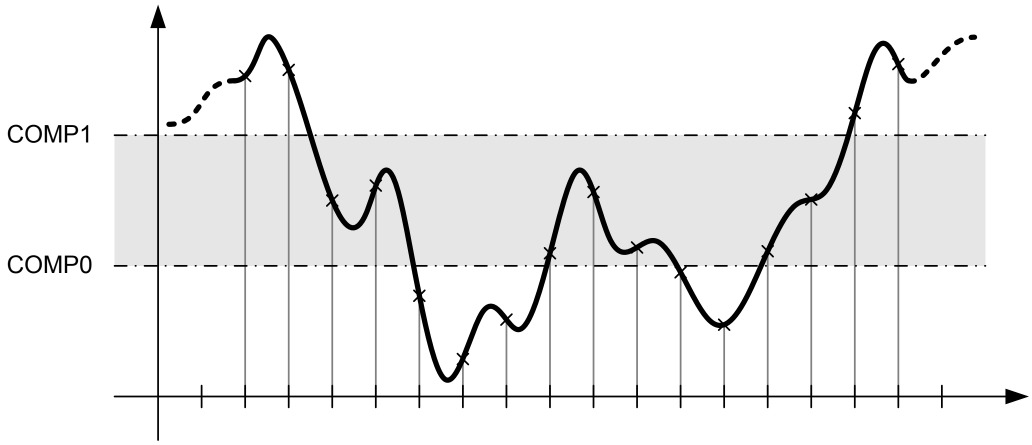


Always	–	0	0	0	0	1	1	1	0	0	1	1	0	0	0	0	1
Once	–	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1
Hysteresis Always	–	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	1
Hysteresis Once	–	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1

Mid-Band Operation

To operate in the mid-band region, either the `CIC` field or the `CTC` field in the **ADCDCCTLn** register must be programmed to 0x1. This setting causes interrupts or triggers to be generated in the mid-band region according the operation mode. Only the Always and Once operational modes are available in the mid-band region. An example of the state of the interrupt/trigger signal in the mid-band region for each of the allowed operational modes is shown in Figure 13-9 on page 419. Note that a "0" in a column following the operational mode name (Always or Once) indicates that the interrupt or trigger signal is de-asserted and a "1" indicates that the signal is asserted.

Figure 13-9. Mid-Band Operation (CIC=0x1 and/or CTC=0x1)

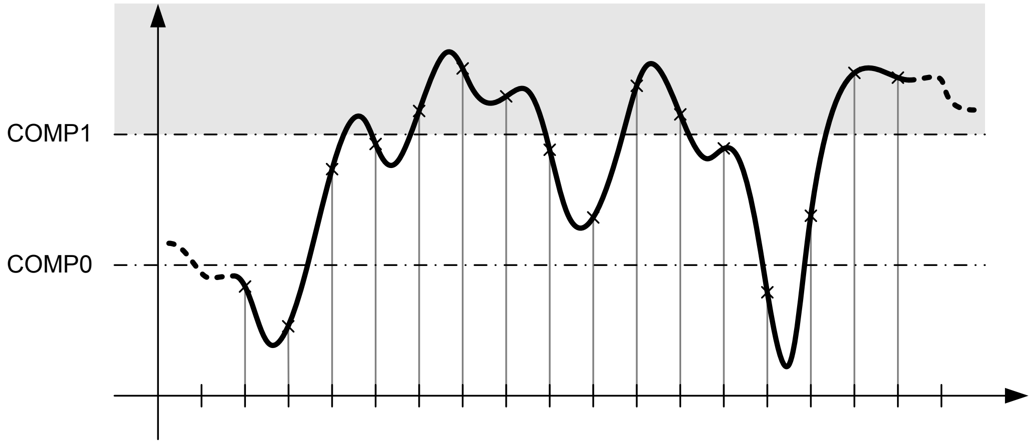


Always	–	0	0	1	1	0	0	0	1	1	1	0	0	1	1	0	0
Once	–	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
Hysteresis Always	–	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Hysteresis Once	–	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

High-Band Operation

To operate in the high-band region, either the `CIC` field or the `CTC` field in the **ADCDCCTLn** register must be programmed to 0x3. This setting causes interrupts or triggers to be generated in the high-band region according the operation mode. An example of the state of the interrupt/trigger signal in the high-band region for each of the allowed operational modes is shown in Figure 13-10 on page 420. Note that a "0" in a column following the operational mode name (Always, Once, Hysteresis Always, and Hysteresis Once) indicates that the interrupt or trigger signal is de-asserted and a "1" indicates that the signal is asserted.

Figure 13-10. High-Band Operation (CIC=0x3 and/or CTC=0x3)



Always	–	0	0	0	0	1	1	1	0	0	1	1	0	0	0	1	1
Once	–	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0
Hysteresis Always	–	0	0	0	0	1	1	1	1	1	1	1	1	0	0	1	1
Hysteresis Once	–	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0

13.4 Initialization and Configuration

In order for the ADC module to be used, the PLL must be enabled and programmed to a supported crystal frequency in the **RCC** register (see page 105). Using unsupported frequencies can cause faulty operation in the ADC module.

13.4.1 Module Initialization

Initialization of the ADC module is a simple process with very few steps: enabling the clock to the ADC, disabling the analog isolation circuit associated with all inputs that are to be used, and reconfiguring the sample sequencer priorities (if needed).

The initialization sequence for the ADC is as follows:

1. Enable the ADC clock by writing a value of 0x0001.0000 to the **RCGC0** register (see page 141).

2. Enable the clock to the appropriate GPIO module via the **RCGC2** register (see page 156). To find out which GPIO port to enable, refer to Table 19-5 on page 640.

3. Set the GPIO `AFSEL` bits for the ADC input pins (see page 307). To determine which GPIOs to configure, see Table 19-4 on page 636.

4. Configure the `PMCn` fields in the **GPIOPCTL** register to assign the `AINx` and `VREFA` signals to the appropriate pins (see page 324 and Table 19-5 on page 640).

5. Disable the analog isolation circuit for all ADC input pins that are to be used by writing a 1 to the appropriate bits of the **GPIOAMSEL** register (see page 323) in the associated GPIO block.

6. If required by the application, reconfigure the sample sequencer priorities in the **ADCSSPRI** register. The default configuration has Sample Sequencer 0 with the highest priority and Sample Sequencer 3 as the lowest priority.

13.4.2 Sample Sequencer Configuration

Configuration of the sample sequencers is slightly more complex than the module initialization because each sample sequencer is completely programmable.

The configuration for each sample sequencer should be as follows:

1. Ensure that the sample sequencer is disabled by clearing the corresponding `ASENn` bit in the **ADCACTSS** register. Programming of the sample sequencers is allowed without having them enabled. Disabling the sequencer during programming prevents erroneous execution if a trigger event were to occur during the configuration process.

2. Configure the trigger event for the sample sequencer in the **ADCEMUX** register.

3. For each sample in the sample sequence, configure the corresponding input source in the **ADCSSMUXn** register.

4. For each sample in the sample sequence, configure the sample control bits in the corresponding nibble in the **ADCSSCTLn** register. When programming the last nibble, ensure that the `END` bit is set. Failure to set the `END` bit causes unpredictable behavior.

5. If interrupts are to be used, set the corresponding `MASK` bit in the **ADCIM** register.

6. Enable the sample sequencer logic by setting the corresponding `ASENn` bit in the **ADCACTSS** register.

13.5 Register Map

Table 13-4 on page 421 lists the ADC registers. The offset listed is a hexadecimal increment to the register's address, relative to the ADC base address of 0x4003.8000.Note that the ADC module clock must be enabled before the registers can be programmed (see page 141).

Table 13-4. ADC Register Map

Offset	Name	Type	Reset	Description	See page
0x000	ADCACTSS	R/W	0x0000.0000	ADC Active Sample Sequencer	424
0x004	ADCRIS	RO	0x0000.0000	ADC Raw Interrupt Status	425
0x008	ADCIM	R/W	0x0000.0000	ADC Interrupt Mask	427
0x00C	ADCISC	R/W1C	0x0000.0000	ADC Interrupt Status and Clear	429
0x010	ADCOSTAT	R/W1C	0x0000.0000	ADC Overflow Status	432
0x014	ADCEMUX	R/W	0x0000.0000	ADC Event Multiplexer Select	434
0x018	ADCUSTAT	R/W1C	0x0000.0000	ADC Underflow Status	437
0x020	ADCSSPRI	R/W	0x0000.3210	ADC Sample Sequencer Priority	438
0x028	ADCPSSI	WO	-	ADC Processor Sample Sequence Initiate	440
0x030	ADCSAC	R/W	0x0000.0000	ADC Sample Averaging Control	442

Table 13-4. ADC Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0x034	ADCDCISC	R/W1C	0x0000.0000	ADC Digital Comparator Interrupt Status and Clear	443
0x038	ADCCTL	R/W	0x0000.0000	ADC Control	445
0x040	ADCSSMUX0	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 0	446
0x044	ADCSSCTL0	R/W	0x0000.0000	ADC Sample Sequence Control 0	448
0x048	ADCSSFIFO0	RO	0x0000.0000	ADC Sample Sequence Result FIFO 0	451
0x04C	ADCSSFSTAT0	RO	0x0000.0100	ADC Sample Sequence FIFO 0 Status	452
0x050	ADCSSOP0	R/W	0x0000.0000	ADC Sample Sequence 0 Operation	454
0x054	ADCSSDC0	R/W	0x0000.0000	ADC Sample Sequence 0 Digital Comparator Select	456
0x060	ADCSSMUX1	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 1	458
0x064	ADCSSCTL1	R/W	0x0000.0000	ADC Sample Sequence Control 1	459
0x068	ADCSSFIFO1	RO	0x0000.0000	ADC Sample Sequence Result FIFO 1	451
0x06C	ADCSSFSTAT1	RO	0x0000.0100	ADC Sample Sequence FIFO 1 Status	452
0x070	ADCSSOP1	R/W	0x0000.0000	ADC Sample Sequence 1 Operation	461
0x074	ADCSSDC1	R/W	0x0000.0000	ADC Sample Sequence 1 Digital Comparator Select	462
0x080	ADCSSMUX2	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 2	458
0x084	ADCSSCTL2	R/W	0x0000.0000	ADC Sample Sequence Control 2	459
0x088	ADCSSFIFO2	RO	0x0000.0000	ADC Sample Sequence Result FIFO 2	451
0x08C	ADCSSFSTAT2	RO	0x0000.0100	ADC Sample Sequence FIFO 2 Status	452
0x090	ADCSSOP2	R/W	0x0000.0000	ADC Sample Sequence 2 Operation	461
0x094	ADCSSDC2	R/W	0x0000.0000	ADC Sample Sequence 2 Digital Comparator Select	462
0x0A0	ADCSSMUX3	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 3	464
0x0A4	ADCSSCTL3	R/W	0x0000.0002	ADC Sample Sequence Control 3	465
0x0A8	ADCSSFIFO3	RO	0x0000.0000	ADC Sample Sequence Result FIFO 3	451
0x0AC	ADCSSFSTAT3	RO	0x0000.0100	ADC Sample Sequence FIFO 3 Status	452
0x0B0	ADCSSOP3	R/W	0x0000.0000	ADC Sample Sequence 3 Operation	466
0x0B4	ADCSSDC3	R/W	0x0000.0000	ADC Sample Sequence 3 Digital Comparator Select	467
0xD00	ADCDCRIC	R/W	0x0000.0000	ADC Digital Comparator Reset Initial Conditions	468
0xE00	ADCDCCTL0	R/W	0x0000.0000	ADC Digital Comparator Control 0	473
0xE04	ADCDCCTL1	R/W	0x0000.0000	ADC Digital Comparator Control 1	473
0xE08	ADCDCCTL2	R/W	0x0000.0000	ADC Digital Comparator Control 2	473
0xE0C	ADCDCCTL3	R/W	0x0000.0000	ADC Digital Comparator Control 3	473
0xE10	ADCDCCTL4	R/W	0x0000.0000	ADC Digital Comparator Control 4	473

Table 13-4. ADC Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xE14	ADCDCCTL5	R/W	0x0000.0000	ADC Digital Comparator Control 5	473
0xE18	ADCDCCTL6	R/W	0x0000.0000	ADC Digital Comparator Control 6	473
0xE1C	ADCDCCTL7	R/W	0x0000.0000	ADC Digital Comparator Control 7	473
0xE40	ADCDCCMP0	R/W	0x0000.0000	ADC Digital Comparator Range 0	477
0xE44	ADCDCCMP1	R/W	0x0000.0000	ADC Digital Comparator Range 1	477
0xE48	ADCDCCMP2	R/W	0x0000.0000	ADC Digital Comparator Range 2	477
0xE4C	ADCDCCMP3	R/W	0x0000.0000	ADC Digital Comparator Range 3	477
0xE50	ADCDCCMP4	R/W	0x0000.0000	ADC Digital Comparator Range 4	477
0xE54	ADCDCCMP5	R/W	0x0000.0000	ADC Digital Comparator Range 5	477
0xE58	ADCDCCMP6	R/W	0x0000.0000	ADC Digital Comparator Range 6	477
0xE5C	ADCDCCMP7	R/W	0x0000.0000	ADC Digital Comparator Range 7	477

13.6 Register Descriptions

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

Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000

This register controls the activation of the sample sequencers. Each sample sequencer can be enabled or disabled independently.

ADC Active Sample Sequencer (ADCACTSS)

Base 0x4003.8000
Offset 0x000
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												ASEN3	ASEN2	ASEN1	ASEN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ASEN3	R/W	0	ADC SS3 Enable Value Description 1 Sample Sequencer 3 is enabled. 0 Sample Sequencer 3 is disabled.
2	ASEN2	R/W	0	ADC SS2 Enable Value Description 1 Sample Sequencer 2 is enabled. 0 Sample Sequencer 2 is disabled.
1	ASEN1	R/W	0	ADC SS1 Enable Value Description 1 Sample Sequencer 1 is enabled. 0 Sample Sequencer 1 is disabled.
0	ASEN0	R/W	0	ADC SS0 Enable Value Description 1 Sample Sequencer 0 is enabled. 0 Sample Sequencer 0 is disabled.

Register 2: ADC Raw Interrupt Status (ADCRIS), offset 0x004

This register shows the status of the raw interrupt signal of each sample sequencer. These bits may be polled by software to look for interrupt conditions without sending the interrupts to the interrupt controller.

ADC Raw Interrupt Status (ADCRIS)

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															INRDC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												INR3	INR2	INR1	INR0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	INRDC	RO	0	Digital Comparator Raw Interrupt Status Value Description 1 At least one bit in the **ADCDCISC** register is set, meaning that a digital comparator interrupt has occurred. 0 All bits in the **ADCDCISC** register are clear.
15:4	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	INR3	RO	0	SS3 Raw Interrupt Status Value Description 1 A sample has completed conversion and the respective **ADCSSCTL3** `IEn` bit is set, enabling a raw interrupt. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the `IN3` bit in the **ADCISC** register.
2	INR2	RO	0	SS2 Raw Interrupt Status Value Description 1 A sample has completed conversion and the respective **ADCSSCTL2** `IEn` bit is set, enabling a raw interrupt. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the `IN2` bit in the **ADCISC** register.

Bit/Field	Name	Type	Reset	Description
1	INR1	RO	0	SS1 Raw Interrupt Status Value Description 1 A sample has completed conversion and the respective **ADCSSCTL1** `IEn` bit is set, enabling a raw interrupt. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the `IN1` bit in the **ADCISC** register.
0	INR0	RO	0	SS0 Raw Interrupt Status Value Description 1 A sample has completed conversion and the respective **ADCSSCTL0** `IEn` bit is set, enabling a raw interrupt. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the `IN0` bit in the **ADCISC** register.

Register 3: ADC Interrupt Mask (ADCIM), offset 0x008

This register controls whether the sample sequencer and digital comparator raw interrupt signals are sent to the interrupt controller. Each raw interrupt signal can be masked independently. Only a single `DCONSSn` bit should be set at any given time. Setting more than one of these bits results in the `INRDC` bit from the **ADCRIS** register being masked, and no interrupt is generated on any of the sample sequencer interrupt lines.

ADC Interrupt Mask (ADCIM)

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												DCONSS3	DCONSS2	DCONSS1	DCONSS0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												MASK3	MASK2	MASK1	MASK0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:20	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	DCONSS3	R/W	0	Digital Comparator Interrupt on SS3 Value Description 1 The raw interrupt signal from the digital comparators (`INRDC` bit in the **ADCRIS** register) is sent to the interrupt controller on the SS3 interrupt line. 0 The status of the digital comparators does not affect the SS3 interrupt status.
18	DCONSS2	R/W	0	Digital Comparator Interrupt on SS2 Value Description 1 The raw interrupt signal from the digital comparators (`INRDC` bit in the **ADCRIS** register) is sent to the interrupt controller on the SS2 interrupt line. 0 The status of the digital comparators does not affect the SS2 interrupt status.
17	DCONSS1	R/W	0	Digital Comparator Interrupt on SS1 Value Description 1 The raw interrupt signal from the digital comparators (`INRDC` bit in the **ADCRIS** register) is sent to the interrupt controller on the SS1 interrupt line. 0 The status of the digital comparators does not affect the SS1 interrupt status.

Bit/Field	Name	Type	Reset	Description
16	DCONSS0	R/W	0	Digital Comparator Interrupt on SS0 Value Description 1 The raw interrupt signal from the digital comparators (`INRDC` bit in the **ADCRIS** register) is sent to the interrupt controller on the SS0 interrupt line. 0 The status of the digital comparators does not affect the SS0 interrupt status.
15:4	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	MASK3	R/W	0	SS3 Interrupt Mask Value Description 1 The raw interrupt signal from Sample Sequencer 3 (**ADCRIS** register `INR3` bit) is sent to the interrupt controller. 0 The status of Sample Sequencer 3 does not affect the SS3 interrupt status.
2	MASK2	R/W	0	SS2 Interrupt Mask Value Description 1 The raw interrupt signal from Sample Sequencer 2 (**ADCRIS** register `INR2` bit) is sent to the interrupt controller. 0 The status of Sample Sequencer 2 does not affect the SS2 interrupt status.
1	MASK1	R/W	0	SS1 Interrupt Mask Value Description 1 The raw interrupt signal from Sample Sequencer 1 (**ADCRIS** register `INR1` bit) is sent to the interrupt controller. 0 The status of Sample Sequencer 1 does not affect the SS1 interrupt status.
0	MASK0	R/W	0	SS0 Interrupt Mask Value Description 1 The raw interrupt signal from Sample Sequencer 0 (**ADCRIS** register `INR0` bit) is sent to the interrupt controller. 0 The status of Sample Sequencer 0 does not affect the SS0 interrupt status.

Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C

This register provides the mechanism for clearing sample sequencer interrupt conditions and shows the status of interrupts generated by the sample sequencers and the digital comparators which have been sent to the interrupt controller. When read, each bit field is the logical AND of the respective `INR` and `MASK` bits. Sample sequencer interrupts are cleared by writing a 1 to the corresponding bit position. Digital comparator interrupts are cleared by writing a 1 to the appropriate bits in the **ADCDCISC** register. If software is polling the **ADCRIS** instead of generating interrupts, the sample sequence `INRn` bits are still cleared via the **ADCISC** register, even if the `INn` bit is not set.

ADC Interrupt Status and Clear (ADCISC)

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												DCINSS3	DCINSS2	DCINSS1	DCINSS0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												IN3	IN2	IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:20	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	DCINSS3	RO	0	Digital Comparator Interrupt Status on SS3 Value Description 1 Both the `INRDC` bit in the **ADCRIS** register and the `DCONSS3` bit in the **ADCIM** register are set, providing a level-base interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1 to it. Clearing this bit also clears the `INRDC` bit in the **ADCRIS** register.
18	DCINSS2	RO	0	Digital Comparator Interrupt Status on SS2 Value Description 1 Both the `INRDC` bit in the **ADCRIS** register and the `DCONSS2` bit in the **ADCIM** register are set, providing a level-base interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1 to it. Clearing this bit also clears the `INRDC` bit in the **ADCRIS** register.

Bit/Field	Name	Type	Reset	Description
17	DCINSS1	RO	0	Digital Comparator Interrupt Status on SS1 Value Description 1 Both the `INRDC` bit in the **ADCRIS** register and the `DCONSS1` bit in the **ADCIM** register are set, providing a level-base interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1 to it. Clearing this bit also clears the `INRDC` bit in the **ADCRIS** register.
16	DCINSS0	RO	0	Digital Comparator Interrupt Status on SS0 Value Description 1 Both the `INRDC` bit in the **ADCRIS** register and the `DCONSS0` bit in the **ADCIM** register are set, providing a level-base interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1 to it. Clearing this bit also clears the `INRDC` bit in the **ADCRIS** register.
15:4	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IN3	R/W1C	0	SS3 Interrupt Status and Clear Value Description 1 Both the `INR3` bit in the **ADCRIS** register and the `MASK3` bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the `INR3` bit in the **ADCRIS** register.
2	IN2	R/W1C	0	SS2 Interrupt Status and Clear Value Description 1 Both the `INR2` bit in the **ADCRIS** register and the `MASK2` bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the `INR2` bit in the **ADCRIS** register.

Bit/Field	Name	Type	Reset	Description
1	IN1	R/W1C	0	SS1 Interrupt Status and Clear Value Description 1 Both the `INR1` bit in the **ADCRIS** register and the `MASK1` bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the `INR1` bit in the **ADCRIS** register.
0	IN0	R/W1C	0	SS0 Interrupt Status and Clear Value Description 1 Both the `INR0` bit in the **ADCRIS** register and the `MASK0` bit in the **ADCIM** register are set, providing a level-based interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the `INR0` bit in the **ADCRIS** register.

Register 5: ADC Overflow Status (ADCOSTAT), offset 0x010

This register indicates overflow conditions in the sample sequencer FIFOs. Once the overflow condition has been handled by software, the condition can be cleared by writing a 1 to the corresponding bit position.

ADC Overflow Status (ADCOSTAT)

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	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OV3	R/W1C	0	SS3 FIFO Overflow Value Description 1 The FIFO for Sample Sequencer 3 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. 0 The FIFO has not overflowed. This bit is cleared by writing a 1.
2	OV2	R/W1C	0	SS2 FIFO Overflow Value Description 1 The FIFO for Sample Sequencer 2 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. 0 The FIFO has not overflowed. This bit is cleared by writing a 1.
1	OV1	R/W1C	0	SS1 FIFO Overflow Value Description 1 The FIFO for Sample Sequencer 1 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. 0 The FIFO has not overflowed. This bit is cleared by writing a 1.

Bit/Field	Name	Type	Reset	Description
0	OV0	R/W1C	0	SS0 FIFO Overflow Value Description 1 The FIFO for Sample Sequencer 0 has hit an overflow condition, meaning that the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. 0 The FIFO has not overflowed. This bit is cleared by writing a 1.

Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014

The **ADCEMUX** selects the event (trigger) that initiates sampling for each sample sequencer. Each sample sequencer can be configured with a unique trigger source.

ADC Event Multiplexer Select (ADCEMUX)

Base 0x4003.8000
Offset 0x014
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	EM3				EM2				EM1				EM0			
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:12	EM3	R/W	0x0	SS3 Trigger Select This field selects the trigger source for Sample Sequencer 3. The valid configurations for this field are:

Value	Event
0x0	Processor (default)
0x1	Analog Comparator 0
0x2	Analog Comparator 1
0x3	reserved
0x4	External (GPIO PB4)
0x5	Timer In addition, the trigger must be enabled with the `TnOTE` bit in the **GPTMCTL** register (see page 358).
0x6	reserved
0x7	reserved
0x8	reserved
0x9	reserved
0xA-0xE	reserved
0xF	Always (continuously sample)

Bit/Field	Name	Type	Reset	Description
11:8	EM2	R/W	0x0	SS2 Trigger Select This field selects the trigger source for Sample Sequencer 2. The valid configurations for this field are: Value Event 0x0 Processor (default) 0x1 Analog Comparator 0 0x2 Analog Comparator 1 0x3 reserved 0x4 External (GPIO PB4) 0x5 Timer In addition, the trigger must be enabled with the `TnOTE` bit in the **GPTMCTL** register (see page 358). 0x6 reserved 0x7 reserved 0x8 reserved 0x9 reserved 0xA-0xE reserved 0xF Always (continuously sample)
7:4	EM1	R/W	0x0	SS1 Trigger Select This field selects the trigger source for Sample Sequencer 1. The valid configurations for this field are: Value Event 0x0 Processor (default) 0x1 Analog Comparator 0 0x2 Analog Comparator 1 0x3 reserved 0x4 External (GPIO PB4) 0x5 Timer In addition, the trigger must be enabled with the `TnOTE` bit in the **GPTMCTL** register (see page 358). 0x6 reserved 0x7 reserved 0x8 reserved 0x9 reserved 0xA-0xE reserved 0xF Always (continuously sample)

Bit/Field	Name	Type	Reset	Description
3:0	EM0	R/W	0x0	SS0 Trigger Select This field selects the trigger source for Sample Sequencer 0 The valid configurations for this field are:

Value	Event
0x0	Processor (default)
0x1	Analog Comparator 0
0x2	Analog Comparator 1
0x3	reserved
0x4	External (GPIO PB4)
0x5	Timer In addition, the trigger must be enabled with the `TnOTE` bit in the **GPTMCTL** register (see page 358).
0x6	reserved
0x7	reserved
0x8	reserved
0x9	reserved
0xA-0xE	reserved
0xF	Always (continuously sample)

Register 7: ADC Underflow Status (ADCUSTAT), offset 0x018

This register indicates underflow conditions in the sample sequencer FIFOs. The corresponding underflow condition is cleared by writing a 1 to the relevant bit position.

ADC Underflow Status (ADCUSTAT)

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	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	UV3	R/W1C	0	SS3 FIFO Underflow The valid configurations for this field are shown below. This bit is cleared by writing a 1. Value Description 1 The FIFO for the Sample Sequencer has hit an underflow condition, meaning that the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. 0 The FIFO has not underflowed.
2	UV2	R/W1C	0	SS2 FIFO Underflow The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.
1	UV1	R/W1C	0	SS1 FIFO Underflow The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.
0	UV0	R/W1C	0	SS0 FIFO Underflow The valid configurations are the same as those for the UV3 field. This bit is cleared by writing a 1.

Register 8: ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020

This register sets the priority for each of the sample sequencers. Out of reset, Sequencer 0 has the highest priority, and Sequencer 3 has the lowest priority. When reconfiguring sequence priorities, each sequence must have a unique priority for the ADC to operate properly.

ADC Sample Sequencer Priority (ADCSSPRI)

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	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13:12	SS3	R/W	0x3	SS3 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 3. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
11:10	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	SS2	R/W	0x2	SS2 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 2. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
7:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:4	SS1	R/W	0x1	SS1 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 1. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
1:0	SS0	R/W	0x0	SS0 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 0. A priority encoding of 0x0 is highest and 0x3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.

Register 9: ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028

This register provides a mechanism for application software to initiate sampling in the sample sequencers. Sample sequences can be initiated individually or in any combination. When multiple sequences are triggered simultaneously, the priority encodings in **ADCSSPRI** dictate execution order.

ADC Processor Sample Sequence Initiate (ADCPSSI)

Base 0x4003.8000
Offset 0x028
Type WO, reset -

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												SS3	SS2	SS1	SS0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	SS3	WO	-	SS3 Initiate Value Description 1 Begin sampling on Sample Sequencer 3, if the sequencer is enabled in the **ADCACTSS** register. 0 No effect. Only a write by software is valid; a read of this register returns no meaningful data.
2	SS2	WO	-	SS2 Initiate Value Description 1 Begin sampling on Sample Sequencer 2, if the sequencer is enabled in the **ADCACTSS** register. 0 No effect. Only a write by software is valid; a read of this register returns no meaningful data.
1	SS1	WO	-	SS1 Initiate Value Description 1 Begin sampling on Sample Sequencer 1, if the sequencer is enabled in the **ADCACTSS** register. 0 No effect. Only a write by software is valid; a read of this register returns no meaningful data.

Bit/Field	Name	Type	Reset	Description
0	SS0	WO	-	SS0 Initiate Value Description 1 Begin sampling on Sample Sequencer 0, if the sequencer is enabled in the **ADCACTSS** register. 0 No effect. Only a write by software is valid; a read of this register returns no meaningful data.

Register 10: ADC Sample Averaging Control (ADCSAC), offset 0x030

This register controls the amount of hardware averaging applied to conversion results. The final conversion result stored in the FIFO is averaged from 2^{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	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	AVG	R/W	0x0	Hardware Averaging Control Specifies the amount of hardware averaging that will be applied to ADC samples. The `AVG` field can be any value between 0 and 6. Entering a value of 7 creates unpredictable results.

Value	Description
0x0	No hardware oversampling
0x1	2x hardware oversampling
0x2	4x hardware oversampling
0x3	8x hardware oversampling
0x4	16x hardware oversampling
0x5	32x hardware oversampling
0x6	64x hardware oversampling
0x7	reserved

Register 11: ADC Digital Comparator Interrupt Status and Clear (ADCDCISC), offset 0x034

This register provides status and acknowledgement of digital comparator interrupts. One bit is provided for each comparator.

ADC Digital Comparator Interrupt Status and Clear (ADCDCISC)

Base 0x4003.8000
Offset 0x034
Type R/W1C, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DCINT7	DCINT6	DCINT5	DCINT4	DCINT3	DCINT2	DCINT1	DCINT0
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	DCINT7	R/W1C	0	Digital Comparator 7 Interrupt Status and Clear Value Description 1 Digital Comparator 7 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
6	DCINT6	R/W1C	0	Digital Comparator 6 Interrupt Status and Clear Value Description 1 Digital Comparator 6 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
5	DCINT5	R/W1C	0	Digital Comparator 5 Interrupt Status and Clear Value Description 1 Digital Comparator 5 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
4	DCINT4	R/W1C	0	Digital Comparator 4 Interrupt Status and Clear Value Description 1 Digital Comparator 4 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.

Bit/Field	Name	Type	Reset	Description
3	DCINT3	R/W1C	0	Digital Comparator 3 Interrupt Status and Clear Value Description 1 Digital Comparator 3 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
2	DCINT2	R/W1C	0	Digital Comparator 2 Interrupt Status and Clear Value Description 1 Digital Comparator 2 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
1	DCINT1	R/W1C	0	Digital Comparator 1 Interrupt Status and Clear Value Description 1 Digital Comparator 1 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.
0	DCINT0	R/W1C	0	Digital Comparator 0 Interrupt Status and Clear Value Description 1 Digital Comparator 0 has generated an interrupt. 0 No interrupt. This bit is cleared by writing a 1.

Register 12: ADC Control (ADCCTL), offset 0x038

This register selects the voltage reference.

ADC Control (ADCCTL)

Base 0x4003.8000
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															VREF
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	VREF	R/W	0	Voltage Reference Select Value Description 1 The external VREFA input is the voltage reference. 0 The internal reference as the voltage reference.

Register 13: ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 0. This register is 32 bits wide and contains information for eight possible samples.

ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0)

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	0x0	8th Sample Input Select The `MUX7` field is used during the eighth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. The value set here indicates the corresponding pin, for example, a value of 0x1 indicates the input is `AIN1`.
27	reserved	RO	0	Software should not rely on the value of 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	0x0	7th Sample Input Select The `MUX6` field is used during the seventh sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
23	reserved	RO	0	Software should not rely on the value of 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	0x0	6th Sample Input Select The `MUX5` field is used during the sixth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
19	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
18:16	MUX4	R/W	0x0	5th Sample Input Select The `MUX4` field is used during the fifth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
15	reserved	RO	0	Software should not rely on the value of 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	0x0	4th Sample Input Select The `MUX3` field is used during the fourth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
11	reserved	RO	0	Software should not rely on the value of 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	0x0	3rd Sample Input Select The `MUX2` field is used during the third sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
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	0x0	2nd Sample Input Select The `MUX1` field is used during the second sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
3	reserved	RO	0	Software should not rely on the value of 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	0x0	1st Sample Input Select The `MUX0` field is used during the first sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.

Register 14: ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044

This register contains the configuration information for each sample for a sequence executed with a sample sequencer. When configuring a sample sequence, the `END` bit must be set for the final sample, whether it be after the first sample, eighth sample, or any sample in between. This register is 32 bits wide and contains information for eight possible samples.

ADC Sample Sequence Control 0 (ADCSSCTL0)

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	8th Sample Temp Sensor Select Value Description 1 The temperature sensor is read during the eighth sample of the sample sequence. 0 The input pin specified by the **ADCSSMUXn** register is read during the eighth sample of the sample sequence.
30	IE7	R/W	0	8th Sample Interrupt Enable Value Description 1 The raw interrupt signal (`INR0` bit) is asserted at the end of the eighth sample's conversion. If the `MASK0` bit in the **ADCIM** register is set, the interrupt is promoted to the interrupt controller. 0 The raw interrupt is not asserted to the interrupt controller. It is legal to have multiple samples within a sequence generate interrupts.
29	END7	R/W	0	8th Sample is End of Sequence Value Description 1 The eighth sample is the last sample of the sequence. 0 Another sample is the sequence is the final sample. It is possible to end the sequence on any sample position. Software must set an `ENDn` bit somewhere within the sequence. Samples defined after the sample containing a set `ENDn` bit are not requested for conversion even though the fields may be non-zero.

Bit/Field	Name	Type	Reset	Description
28	D7	R/W	0	8th Sample Diff Input Select Value Description 1 The analog input is differentially sampled. The corresponding **ADCSSMUXn** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". 0 The analog inputs are not differentially sampled. Because the temperature sensor does not have a differential option, this bit must not be set when the `TS7` bit is set.
27	TS6	R/W	0	7th Sample Temp Sensor Select Same definition as `TS7` but used during the seventh sample.
26	IE6	R/W	0	7th Sample Interrupt Enable Same definition as `IE7` but used during the seventh sample.
25	END6	R/W	0	7th Sample is End of Sequence Same definition as `END7` but used during the seventh sample.
24	D6	R/W	0	7th Sample Diff Input Select Same definition as `D7` but used during the seventh sample.
23	TS5	R/W	0	6th Sample Temp Sensor Select Same definition as `TS7` but used during the sixth sample.
22	IE5	R/W	0	6th Sample Interrupt Enable Same definition as `IE7` but used during the sixth sample.
21	END5	R/W	0	6th Sample is End of Sequence Same definition as `END7` but used during the sixth sample.
20	D5	R/W	0	6th Sample Diff Input Select Same definition as `D7` but used during the sixth sample.
19	TS4	R/W	0	5th Sample Temp Sensor Select Same definition as `TS7` but used during the fifth sample.
18	IE4	R/W	0	5th Sample Interrupt Enable Same definition as `IE7` but used during the fifth sample.
17	END4	R/W	0	5th Sample is End of Sequence Same definition as `END7` but used during the fifth sample.
16	D4	R/W	0	5th Sample Diff Input Select Same definition as `D7` but used during the fifth sample.
15	TS3	R/W	0	4th Sample Temp Sensor Select Same definition as `TS7` but used during the fourth sample.

Bit/Field	Name	Type	Reset	Description
14	IE3	R/W	0	4th Sample Interrupt Enable Same definition as `IE7` but used during the fourth sample.
13	END3	R/W	0	4th Sample is End of Sequence Same definition as `END7` but used during the fourth sample.
12	D3	R/W	0	4th Sample Diff Input Select Same definition as `D7` but used during the fourth sample.
11	TS2	R/W	0	3rd Sample Temp Sensor Select Same definition as `TS7` but used during the third sample.
10	IE2	R/W	0	3rd Sample Interrupt Enable Same definition as `IE7` but used during the third sample.
9	END2	R/W	0	3rd Sample is End of Sequence Same definition as `END7` but used during the third sample.
8	D2	R/W	0	3rd Sample Diff Input Select Same definition as `D7` but used during the third sample.
7	TS1	R/W	0	2nd Sample Temp Sensor Select Same definition as `TS7` but used during the second sample.
6	IE1	R/W	0	2nd Sample Interrupt Enable Same definition as `IE7` but used during the second sample.
5	END1	R/W	0	2nd Sample is End of Sequence Same definition as `END7` but used during the second sample.
4	D1	R/W	0	2nd Sample Diff Input Select Same definition as `D7` but used during the second sample.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as `TS7` but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as `IE7` but used during the first sample.
1	END0	R/W	0	1st Sample is End of Sequence Same definition as `END7` but used during the first sample.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as `D7` but used during the first sample.

Register 15: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048

Register 16: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068

Register 17: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088

Register 18: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8

Important: Use caution when reading this register. Performing a read may change bit status.

This register contains the conversion results for samples collected with the sample sequencer (the **ADCSSFIFO0** register is used for Sample Sequencer 0, **ADCSSFIFO1** for Sequencer 1, **ADCSSFIFO2** for Sequencer 2, and **ADCSSFIFO3** for Sequencer 3). Reads of this register return conversion result data in the order sample 0, sample 1, and so on, until the FIFO is empty. If the FIFO is not properly handled by software, overflow and underflow conditions are registered in the **ADCOSTAT** and **ADCUSTAT** registers.

ADC Sample Sequence Result FIFO 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	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:0	DATA	RO	0x000	Conversion Result Data

Register 19: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C

Register 20: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C

Register 21: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C

Register 22: ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC

This register provides a window into the sample sequencer, providing full/empty status information as well as the positions of the head and tail pointers. The reset value of 0x100 indicates an empty FIFO. The **ADCSSFSTAT0** register provides status on FIFO0, which has 8 entries; **ADCSSFSTAT1** on FIFO1, which has 4 entries; **ADCSSFSTAT2** on FIFO2, which has 4 entries; and **ADCSSFSTAT3** on FIFO3 which has a single entry.

ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0)

Base 0x4003.8000
Offset 0x04C
Type RO, reset 0x0000.0100

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			FULL	reserved			EMPTY	HPTR				TPTR			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	FULL	RO	0	FIFO Full Value Description 1 The FIFO is currently full. 0 The FIFO is not currently full.
11:9	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	EMPTY	RO	1	FIFO Empty Value Description 1 The FIFO is currently empty. 0 The FIFO is not currently empty.

Bit/Field	Name	Type	Reset	Description
7:4	HPTR	RO	0x0	FIFO Head Pointer This field contains the current "head" pointer index for the FIFO, that is, the next entry to be written.
3:0	TPTR	RO	0x0	FIFO Tail Pointer This field contains the current "tail" pointer index for the FIFO, that is, the next entry to be read.

Register 23: ADC Sample Sequence 0 Operation (ADCSSOP0), offset 0x050

This register determines whether the sample from the given conversion on Sample Sequence 0 is saved in the Sample Sequence FIFO0 or sent to the digital comparator unit.

ADC Sample Sequence 0 Operation (ADCSSOP0)

Base 0x4003.8000
Offset 0x050
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved			S7DCOP	reserved			S6DCOP	reserved			S5DCOP	reserved			S4DCOP
Type	RO	RO	RO	R/W	RO	RO	RO	R/W	RO	RO	RO	R/W	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			S3DCOP	reserved			S2DCOP	reserved			S1DCOP	reserved			S0DCOP
Type	RO	RO	RO	R/W	RO	RO	RO	R/W	RO	RO	RO	R/W	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	S7DCOP	R/W	0	Sample 7 Digital Comparator Operation Value Description 1 The eighth sample is sent to the digital comparator unit specified by the `S7DCSEL` bit in the **ADCSSDC0** register, and the value is not written to the FIFO. 0 The eighth sample is saved in Sample Sequence FIFO0.
27:25	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
24	S6DCOP	R/W	0	Sample 6 Digital Comparator Operation Same definition as `S7DCOP` but used during the seventh sample.
23:21	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
20	S5DCOP	R/W	0	Sample 5 Digital Comparator Operation Same definition as `S7DCOP` but used during the sixth sample.
19:17	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	S4DCOP	R/W	0	Sample 4 Digital Comparator Operation Same definition as `S7DCOP` but used during the fifth sample.
15:13	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
12	S3DCOP	R/W	0	Sample 3 Digital Comparator Operation Same definition as `S7DCOP` but used during the fourth sample.
11:9	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	S2DCOP	R/W	0	Sample 2 Digital Comparator Operation Same definition as `S7DCOP` but used during the third sample.
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	S1DCOP	R/W	0	Sample 1 Digital Comparator Operation Same definition as `S7DCOP` but used during the second sample.
3:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	S0DCOP	R/W	0	Sample 0 Digital Comparator Operation Same definition as `S7DCOP` but used during the first sample.

Register 24: ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0), offset 0x054

This register determines which digital comparator receives the sample from the given conversion on Sample Sequence 0, if the corresponding `SnDCOP` bit in the **ADCSSOP0** register is set.

ADC Sample Sequence 0 Digital Comparator Select (ADCSSDC0)

Base 0x4003.8000
Offset 0x054
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	S7DCSEL				S6DCSEL				S5DCSEL				S4DCSEL			
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL			
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:28	S7DCSEL	R/W	0x0	Sample 7 Digital Comparator Select When the `S7DCOP` bit in the **ADCSSOP0** register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the eighth sample from Sample Sequencer 0. **Note:** Values not listed are reserved. Value Description 0x0 Digital Comparator Unit 0 (**ADCDCCMP0** and **ADCCCTL0**) 0x1 Digital Comparator Unit 1 (**ADCDCCMP1** and **ADCCCTL1**) 0x2 Digital Comparator Unit 2 (**ADCDCCMP2** and **ADCCCTL2**) 0x3 Digital Comparator Unit 3 (**ADCDCCMP3** and **ADCCCTL3**) 0x4 Digital Comparator Unit 4 (**ADCDCCMP4** and **ADCCCTL4**) 0x5 Digital Comparator Unit 5 (**ADCDCCMP5** and **ADCCCTL5**) 0x6 Digital Comparator Unit 6 (**ADCDCCMP6** and **ADCCCTL6**) 0x7 Digital Comparator Unit 7 (**ADCDCCMP7** and **ADCCCTL7**)
27:24	S6DCSEL	R/W	0x0	Sample 6 Digital Comparator Select This field has the same encodings as `S7DCSEL` but is used during the seventh sample.
23:20	S5DCSEL	R/W	0x0	Sample 5 Digital Comparator Select This field has the same encodings as `S7DCSEL` but is used during the sixth sample.
19:16	S4DCSEL	R/W	0x0	Sample 4 Digital Comparator Select This field has the same encodings as `S7DCSEL` but is used during the fifth sample.

Bit/Field	Name	Type	Reset	Description
15:12	S3DCSEL	R/W	0x0	Sample 3 Digital Comparator Select This field has the same encodings as `S7DCSEL` but is used during the fourth sample.
11:8	S2DCSEL	R/W	0x0	Sample 2 Digital Comparator Select This field has the same encodings as `S7DCSEL` but is used during the third sample.
7:4	S1DCSEL	R/W	0x0	Sample 1 Digital Comparator Select This field has the same encodings as `S7DCSEL` but is used during the second sample.
3:0	S0DCSEL	R/W	0x0	Sample 0 Digital Comparator Select This field has the same encodings as `S7DCSEL` but is used during the first sample.

Register 25: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060

Register 26: ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 1 or 2. These registers are 16 bits wide and contain information for four possible samples. See the **ADCSSMUX0** register on page 446 for detailed bit descriptions. The **ADCSSMUX1** register affects Sample Sequencer 1 and the **ADCSSMUX2** register affects Sample Sequencer 2.

ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1)

Base 0x4003.8000
Offset 0x060
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	MUX3			reserved	MUX2			reserved	MUX1			reserved	MUX0		
Type	RO	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	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	MUX3	R/W	0x0	4th Sample Input Select
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	0x0	3rd Sample Input Select
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	0x0	2nd Sample Input Select
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	0x0	1st Sample Input Select

Register 27: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064

Register 28: ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084

These registers contain the configuration information for each sample for a sequence executed with Sample Sequencer 1 or 2. When configuring a sample sequence, the `END` bit must be set for the final sample, whether it be after the first sample, fourth sample, or any sample in between. These registers are 16-bits wide and contain information for four possible samples. See the **ADCSSCTL0** register on page 448 for detailed bit descriptions. The **ADCSSCTL1** register configures Sample Sequencer 1 and the **ADCSSCTL2** register configures Sample Sequencer 2.

ADC Sample Sequence Control 1 (ADCSSCTL1)

Base 0x4003.8000
Offset 0x064
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	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	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	TS3	R/W	0	4th Sample Temp Sensor Select Same definition as `TS7` but used during the fourth sample.
14	IE3	R/W	0	4th Sample Interrupt Enable Same definition as `IE7` but used during the fourth sample.
13	END3	R/W	0	4th Sample is End of Sequence Same definition as `END7` but used during the fourth sample.
12	D3	R/W	0	4th Sample Diff Input Select Same definition as `D7` but used during the fourth sample.
11	TS2	R/W	0	3rd Sample Temp Sensor Select Same definition as `TS7` but used during the third sample.
10	IE2	R/W	0	3rd Sample Interrupt Enable Same definition as `IE7` but used during the third sample.
9	END2	R/W	0	3rd Sample is End of Sequence Same definition as `END7` but used during the third sample.
8	D2	R/W	0	3rd Sample Diff Input Select Same definition as `D7` but used during the third sample.

Bit/Field	Name	Type	Reset	Description
7	TS1	R/W	0	2nd Sample Temp Sensor Select Same definition as `TS7` but used during the second sample.
6	IE1	R/W	0	2nd Sample Interrupt Enable Same definition as `IE7` but used during the second sample.
5	END1	R/W	0	2nd Sample is End of Sequence Same definition as `END7` but used during the second sample.
4	D1	R/W	0	2nd Sample Diff Input Select Same definition as `D7` but used during the second sample.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as `TS7` but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as `IE7` but used during the first sample.
1	END0	R/W	0	1st Sample is End of Sequence Same definition as `END7` but used during the first sample.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as `D7` but used during the first sample.

Register 29: ADC Sample Sequence 1 Operation (ADCSSOP1), offset 0x070

Register 30: ADC Sample Sequence 2 Operation (ADCSSOP2), offset 0x090

This register determines whether the sample from the given conversion on Sample Sequence n is saved in the Sample Sequence n FIFO or sent to the digital comparator unit. The **ADCSSOP1** register controls Sample Sequencer 1 and the **ADCSSOP2** register controls Sample Sequencer 2.

ADC Sample Sequence 1 Operation (ADCSSOP1)

Base 0x4003.8000
Offset 0x070
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			S3DCOP	reserved			S2DCOP	reserved			S1DCOP	reserved			S0DCOP
Type	RO	RO	RO	R/W	RO	RO	RO	R/W	RO	RO	RO	R/W	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	S3DCOP	R/W	0	Sample 3 Digital Comparator Operation Value Description 1 The fourth sample is sent to the digital comparator unit specified by the `S3DCSEL` bit in the **ADCSSDC0n** register, and the value is not written to the FIFO. 0 The fourth sample is saved in Sample Sequence FIFOn.
11:9	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	S2DCOP	R/W	0	Sample 2 Digital Comparator Operation Same definition as `S3DCOP` but used during the third sample.
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	S1DCOP	R/W	0	Sample 1 Digital Comparator Operation Same definition as `S3DCOP` but used during the second sample.
3:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	S0DCOP	R/W	0	Sample 0 Digital Comparator Operation Same definition as `S3DCOP` but used during the first sample.

Register 31: ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1), offset 0x074

Register 32: ADC Sample Sequence 2 Digital Comparator Select (ADCSSDC2), offset 0x094

These registers determine which digital comparator receives the sample from the given conversion on Sample Sequence n if the corresponding `SnDCOP` bit in the **ADCSSOPn** register is set. The **ADCSSDC1** register controls the selection for Sample Sequencer 1 and the **ADCSSDC2** register controls the selection for Sample Sequencer 2.

ADC Sample Sequence 1 Digital Comparator Select (ADCSSDC1)

Base 0x4003.8000
Offset 0x074
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL			
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:12	S3DCSEL	R/W	0x0	Sample 3 Digital Comparator Select When the `S3DCOP` bit in the **ADCSSOPn** register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the eighth sample from Sample Sequencer n. **Note:** Values not listed are reserved. Value Description 0x0 Digital Comparator Unit 0 (**ADCDCCMP0** and **ADCCCTL0**) 0x1 Digital Comparator Unit 1 (**ADCDCCMP1** and **ADCCCTL1**) 0x2 Digital Comparator Unit 2 (**ADCDCCMP2** and **ADCCCTL2**) 0x3 Digital Comparator Unit 3 (**ADCDCCMP3** and **ADCCCTL3**) 0x4 Digital Comparator Unit 4 (**ADCDCCMP4** and **ADCCCTL4**) 0x5 Digital Comparator Unit 5 (**ADCDCCMP5** and **ADCCCTL5**) 0x6 Digital Comparator Unit 6 (**ADCDCCMP6** and **ADCCCTL6**) 0x7 Digital Comparator Unit 7 (**ADCDCCMP7** and **ADCCCTL7**)
11:8	S2DCSEL	R/W	0x0	Sample 2 Digital Comparator Select This field has the same encodings as `S3DCSEL` but is used during the third sample.

Bit/Field	Name	Type	Reset	Description
7:4	S1DCSEL	R/W	0x0	Sample 1 Digital Comparator Select This field has the same encodings as `S3DCSEL` but is used during the second sample.
3:0	S0DCSEL	R/W	0x0	Sample 0 Digital Comparator Select This field has the same encodings as `S3DCSEL` but is used during the first sample.

Register 33: ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0

This register defines the analog input configuration for the sample executed with Sample Sequencer 3. This register is 4 bits wide and contains information for one possible sample. See the **ADCSSMUX0** register on page 446 for detailed bit descriptions.

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	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	MUX0	R/W	0	1st Sample Input Select

Register 34: ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4

This register contains the configuration information for a sample executed with Sample Sequencer 3. The `END0` bit is always set as this sequencer can execute only one sample. This register is 4 bits wide and contains information for one possible sample. See the **ADCSSCTL0** register on page 448 for detailed bit descriptions.

ADC Sample Sequence Control 3 (ADCSSCTL3)

Base 0x4003.8000
Offset 0x0A4
Type R/W, reset 0x0000.0002

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												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	1	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as `TS7` but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as `IE7` but used during the first sample.
1	END0	R/W	1	1st Sample is End of Sequence Same definition as `END7` but used during the first sample. Because this sequencer has only one entry, this bit must be set.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as `D7` but used during the first sample.

Register 35: ADC Sample Sequence 3 Operation (ADCSSOP3), offset 0x0B0

This register determines whether the sample from the given conversion on Sample Sequence 3 is saved in the Sample Sequence 3 FIFO or sent to the digital comparator unit.

ADC Sample Sequence 3 Operation (ADCSSOP3)

Base 0x4003.8000
Offset 0x0B0
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															S0DCOP
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	S0DCOP	R/W	0	Sample 0 Digital Comparator Operation

Value	Description
1	The sample is sent to the digital comparator unit specified by the `S0DCSEL` bit in the **ADCSSDC03** register, and the value is not written to the FIFO.
0	The sample is saved in Sample Sequence FIFO3.

Register 36: ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3), offset 0x0B4

This register determines which digital comparator receives the sample from the given conversion on Sample Sequence 3 if the corresponding `SnDCOP` bit in the **ADCSSOP3** register is set.

ADC Sample Sequence 3 Digital Comparator Select (ADCSSDC3)

Base 0x4003.8000
Offset 0x0B4
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												S0DCSEL			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	S0DCSEL	R/W	0x0	Sample 0 Digital Comparator Select When the `S0DCOP` bit in the **ADCSSOP3** register is set, this field indicates which digital comparator unit (and its associated set of control registers) receives the sample from Sample Sequencer 3. **Note:** Values not listed are reserved.

Value	Description
0x0	Digital Comparator Unit 0 (**ADCDCCMP0** and **ADCCCTL0**)
0x1	Digital Comparator Unit 1 (**ADCDCCMP1** and **ADCCCTL1**)
0x2	Digital Comparator Unit 2 (**ADCDCCMP2** and **ADCCCTL2**)
0x3	Digital Comparator Unit 3 (**ADCDCCMP3** and **ADCCCTL3**)
0x4	Digital Comparator Unit 4 (**ADCDCCMP4** and **ADCCCTL4**)
0x5	Digital Comparator Unit 5 (**ADCDCCMP5** and **ADCCCTL5**)
0x6	Digital Comparator Unit 6 (**ADCDCCMP6** and **ADCCCTL6**)
0x7	Digital Comparator Unit 7 (**ADCDCCMP7** and **ADCCCTL7**)

Register 37: ADC Digital Comparator Reset Initial Conditions (ADCDCRIC), offset 0xD00

This register provides the ability to reset any of the digital comparator interrupt or trigger functions back to their initial conditions. Resetting these functions ensures that the data that is being used by the interrupt and trigger functions in the digital comparator unit is not stale.

ADC Digital Comparator Reset Initial Conditions (ADCDCRIC)

Base 0x4003.8000
Offset 0xD00
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved								DCTRIG7	DCTRIG6	DCTRIG5	DCTRIG4	DCTRIG3	DCTRIG2	DCTRIG1	DCTRIG0
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DCINT7	DCINT6	DCINT5	DCINT4	DCINT3	DCINT2	DCINT1	DCINT0
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:24	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23	DCTRIG7	R/W	0	Digital Comparator Trigger 7 Value Description 1 Resets the Digital Comparator 7 trigger unit to its initial conditions. 0 No effect. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
22	DCTRIG6	R/W	0	Digital Comparator Trigger 6 Value Description 1 Resets the Digital Comparator 6 trigger unit to its initial conditions. 0 No effect. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

Bit/Field	Name	Type	Reset	Description
21	DCTRIG5	R/W	0	Digital Comparator Trigger 5 Value Description 1 Resets the Digital Comparator 5 trigger unit to its initial conditions. 0 No effect. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
20	DCTRIG4	R/W	0	Digital Comparator Trigger 4 Value Description 1 Resets the Digital Comparator 4 trigger unit to its initial conditions. 0 No effect. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
19	DCTRIG3	R/W	0	Digital Comparator Trigger 3 Value Description 1 Resets the Digital Comparator 3 trigger unit to its initial conditions. 0 No effect. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
18	DCTRIG2	R/W	0	Digital Comparator Trigger 2 Value Description 1 Resets the Digital Comparator 2 trigger unit to its initial conditions. 0 No effect. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

Bit/Field	Name	Type	Reset	Description
17	DCTRIG1	R/W	0	Digital Comparator Trigger 1 Value Description 1 Resets the Digital Comparator 1 trigger unit to its initial conditions. 0 No effect. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
16	DCTRIG0	R/W	0	Digital Comparator Trigger 0 Value Description 1 Resets the Digital Comparator 0 trigger unit to its initial conditions. 0 No effect. When the trigger has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the trigger, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
15:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	DCINT7	R/W	0	Digital Comparator Interrupt 7 Value Description 1 Resets the Digital Comparator 7 interrupt unit to its initial conditions. 0 No effect. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

Bit/Field	Name	Type	Reset	Description
6	DCINT6	R/W	0	Digital Comparator Interrupt 6 Value Description 1 Resets the Digital Comparator 6 interrupt unit to its initial conditions. 0 No effect. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
5	DCINT5	R/W	0	Digital Comparator Interrupt 5 Value Description 1 Resets the Digital Comparator 5 interrupt unit to its initial conditions. 0 No effect. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
4	DCINT4	R/W	0	Digital Comparator Interrupt 4 Value Description 1 Resets the Digital Comparator 4 interrupt unit to its initial conditions. 0 No effect. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
3	DCINT3	R/W	0	Digital Comparator Interrupt 3 Value Description 1 Resets the Digital Comparator 3 interrupt unit to its initial conditions. 0 No effect. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

Bit/Field	Name	Type	Reset	Description
2	DCINT2	R/W	0	Digital Comparator Interrupt 2 Value Description 1 Resets the Digital Comparator 2 interrupt unit to its initial conditions. 0 No effect. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
1	DCINT1	R/W	0	Digital Comparator Interrupt 1 Value Description 1 Resets the Digital Comparator 1 interrupt unit to its initial conditions. 0 No effect. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.
0	DCINT0	R/W	0	Digital Comparator Interrupt 0 Value Description 1 Resets the Digital Comparator 0 interrupt unit to its initial conditions. 0 No effect. When the interrupt has been cleared, this bit is automatically cleared. Because the digital comparators use the current and previous ADC conversion values to determine when to assert the interrupt, it is important to reset the digital comparator to initial conditions when starting a new sequence so that stale data is not used.

Register 38: ADC Digital Comparator Control 0 (ADCDCCTL0), offset 0xE00

Register 39: ADC Digital Comparator Control 1 (ADCDCCTL1), offset 0xE04

Register 40: ADC Digital Comparator Control 2 (ADCDCCTL2), offset 0xE08

Register 41: ADC Digital Comparator Control 3 (ADCDCCTL3), offset 0xE0C

Register 42: ADC Digital Comparator Control 4 (ADCDCCTL4), offset 0xE10

Register 43: ADC Digital Comparator Control 5 (ADCDCCTL5), offset 0xE14

Register 44: ADC Digital Comparator Control 6 (ADCDCCTL6), offset 0xE18

Register 45: ADC Digital Comparator Control 7 (ADCDCCTL7), offset 0xE1C

This register provides the comparison encodings that generate an interrupt or PWM trigger.

ADC Digital Comparator Control 0 (ADCDCCTL0)

Base 0x4003.8000
Offset 0xE00
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved			CTE	CTC		CTM		reserved			CIE	CIC		CIM	
Type	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	RO	RO	RO	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	CTE	R/W	0	Comparison Trigger Enable Value Description 1 Enables the trigger function state machine. The ADC conversion data is used to determine if a trigger should be generated according to the programming of the `CTC` and `CTM` fields. 0 Disables the trigger function state machine. ADC conversion data is ignored by the trigger function.

Bit/Field	Name	Type	Reset	Description
11:10	CTC	R/W	0x0	Comparison Trigger Condition This field specifies the operational region in which a trigger is generated when the ADC conversion data is compared against the values of `COMP0` and `COMP1`. The `COMP0` and `COMP1` fields are defined in the **ADCDCCMPx** registers. Value Description 0x0 Low Band ADC Data < `COMP0` and < `COMP1` 0x1 Mid Band `COMP0` ≤ ADC Data < `COMP1` 0x2 reserved 0x3 High Band `COMP0` ≤ `COMP1` ≤ ADC Data
9:8	CTM	R/W	0x0	Comparison Trigger Mode This field specifies the mode by which the trigger comparison is made. Value Description 0x0 Always This mode generates a trigger every time the ADC conversion data falls within the selected operational region. 0x1 Once This mode generates a trigger the first time that the ADC conversion data enters the selected operational region. 0x2 Hysteresis Always This mode generates a trigger when the ADC conversion data falls within the selected operational region and continues to generate the trigger until the hysteresis condition is cleared by entering the opposite operational region. Note that the hysteresis modes are only defined for `CTC` encodings of 0x0 and 0x3. 0x3 Hysteresis Once This mode generates a trigger the first time that the ADC conversion data falls within the selected operational region. No additional triggers are generated until the hysteresis condition is cleared by entering the opposite operational region. Note that the hysteresis modes are only defined for `CTC` encodings of 0x0 and 0x3.
7:5	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
4	CIE	R/W	0	Comparison Interrupt Enable Value Description 1 Enables the comparison interrupt. The ADC conversion data is used to determine if an interrupt should be generated according to the programming of the `CIC` and `CIM` fields. 0 Disables the comparison interrupt. ADC conversion data has no effect on interrupt generation.
3:2	CIC	R/W	0x0	Comparison Interrupt Condition This field specifies the operational region in which an interrupt is generated when the ADC conversion data is compared against the values of `COMP0` and `COMP1`. The `COMP0` and `COMP1` fields are defined in the **ADCDCCMPx** registers. Value Description 0x0 Low Band ADC Data < `COMP0` and < `COMP1` 0x1 Mid Band `COMP0` ≤ ADC Data < `COMP1` 0x2 reserved 0x3 High Band `COMP0` < `COMP1` ≤ ADC Data

Bit/Field	Name	Type	Reset	Description
1:0	CIM	R/W	0x0	Comparison Interrupt Mode This field specifies the mode by which the interrupt comparison is made. Value Description 0x0 Always This mode generates an interrupt every time the ADC conversion data falls within the selected operational region. 0x1 Once This mode generates an interrupt the first time that the ADC conversion data enters the selected operational region. 0x2 Hysteresis Always This mode generates an interrupt when the ADC conversion data falls within the selected operational region and continues to generate the interrupt until the hysteresis condition is cleared by entering the opposite operational region. Note that the hysteresis modes are only defined for `CTC` encodings of 0x0 and 0x3. 0x3 Hysteresis Once This mode generates an interrupt the first time that the ADC conversion data falls within the selected operational region. No additional interrupts are generated until the hysteresis condition is cleared by entering the opposite operational region. Note that the hysteresis modes are only defined for `CTC` encodings of 0x0 and 0x3.

Register 46: ADC Digital Comparator Range 0 (ADCDCCMP0), offset 0xE40

Register 47: ADC Digital Comparator Range 1 (ADCDCCMP1), offset 0xE44

Register 48: ADC Digital Comparator Range 2 (ADCDCCMP2), offset 0xE48

Register 49: ADC Digital Comparator Range 3 (ADCDCCMP3), offset 0xE4C

Register 50: ADC Digital Comparator Range 4 (ADCDCCMP4), offset 0xE50

Register 51: ADC Digital Comparator Range 5 (ADCDCCMP5), offset 0xE54

Register 52: ADC Digital Comparator Range 6 (ADCDCCMP6), offset 0xE58

Register 53: ADC Digital Comparator Range 7 (ADCDCCMP7), offset 0xE5C

This register defines the comparison values that are used to determine if the ADC conversion data falls in the appropriate operating region. Note that the value in the `COMP1` field must be greater than or equal to the value in the `COMP0` field or unexpected results can occur.

ADC Digital Comparator Range 0 (ADCDCCMP0)

Base 0x4003.8000
Offset 0xE40
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved						COMP1									
Type	RO	RO	RO	RO	RO	RO	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
	reserved						COMP0									
Type	RO	RO	RO	RO	RO	RO	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
31:26	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25:16	COMP1	R/W	0x000	Compare 1 The value in this field is compared against the ADC conversion data. The result of the comparison is used to determine if the data lies within the high-band region. Note that the value of `COMP1` must be greater than or equal to the value of `COMP0`.
15:10	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:0	COMP0	R/W	0x000	Compare 0 The value in this field is compared against the ADC conversion data. The result of the comparison is used to determine if the data lies within the low-band region.

14 Universal Asynchronous Receivers/Transmitters (UARTs)

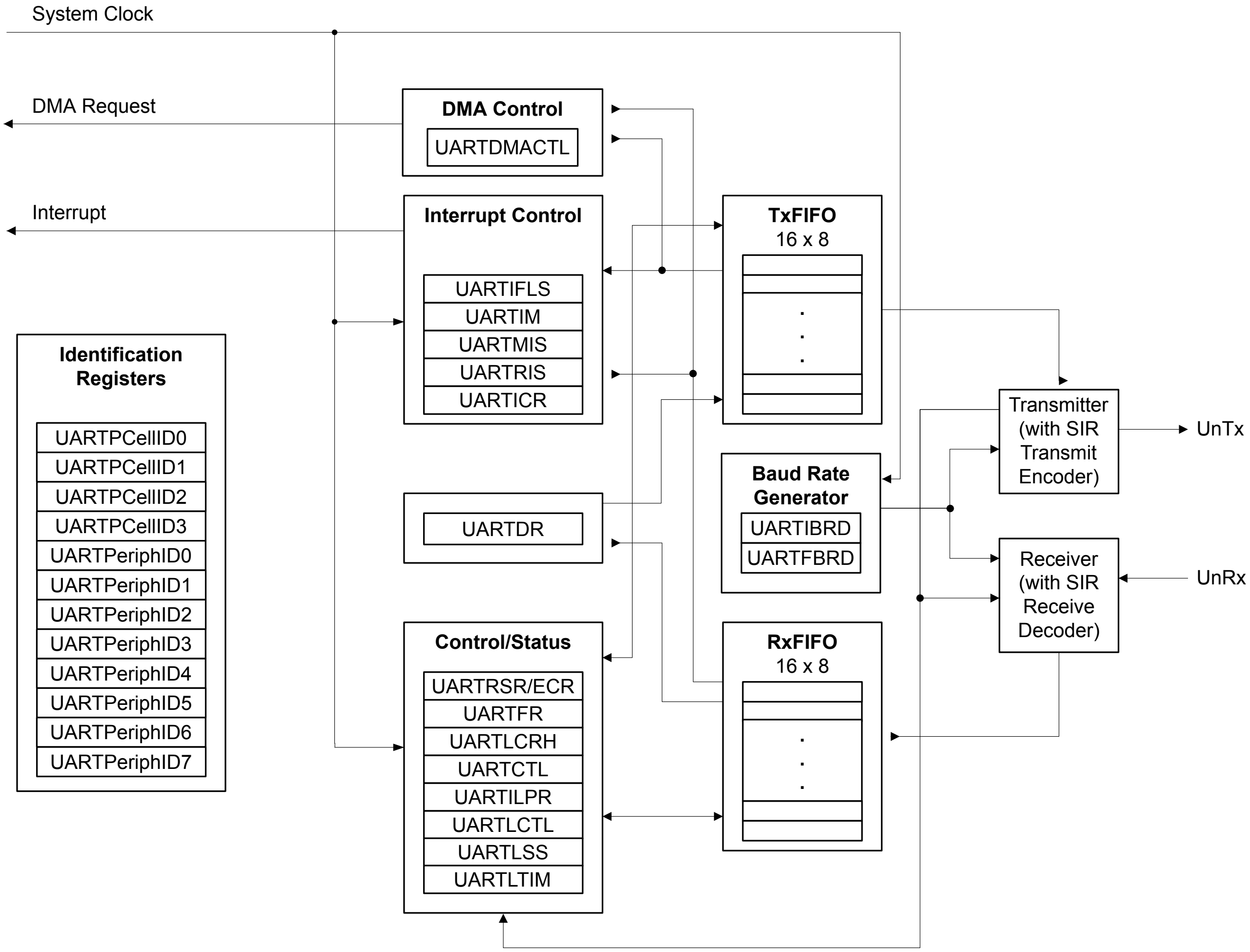
The Stellaris® LM3S1W16 controller includes three Universal Asynchronous Receiver/Transmitter (UART) with the following features:

- Programmable baud-rate generator allowing speeds up to 3.125 Mbps for regular speed (divide by 16) and 6.25 Mbps for high speed (divide by 8)
- Separate 16x8 transmit (TX) and receive (RX) FIFOs to reduce CPU interrupt service loading
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
- Standard asynchronous communication bits for start, stop, and parity
- 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 μs) bit durations
  - Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration
- Support for communication with ISO 7816 smart cards
- LIN protocol support
- Standard FIFO-level and End-of-Transmission interrupts
- Efficient transfers using Micro Direct Memory Access Controller (μDMA)
  - Separate channels for transmit and receive
  - Receive single request asserted when data is in the FIFO; burst request asserted at programmed FIFO level

  - Transmit single request asserted when there is space in the FIFO; burst request asserted at programmed FIFO level

14.1 Block Diagram

Figure 14-1. UART Module Block Diagram



14.2 Signal Description

Table 14-1 on page 480 lists the external signals of the UART module and describes the function of each. The UART signals are alternate functions for some GPIO signals and default to be GPIO signals at reset, with the exception of the `U0Rx` and `U0Tx` pins which default to the UART function. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for these UART signals. The `AFSEL` bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 307) should be set to choose the UART function. The number in parentheses is the encoding that must be programmed into the `PMCn` field in the **GPIO Port Control (GPIOPCTL)** register (page 324) to assign the UART signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 285.

Table 14-1. Signals for UART

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`U0Rx`	17	PA0 (1)	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
`U0Tx`	18	PA1 (1)	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
`U1Rx`	15 17 41 58 61 63	PC6 (5) PA0 (9) PB0 (5) PB4 (7) PD0 (5) PD2 (1)	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
`U1Tx`	16 18 42 57 62 64	PC7 (5) PA1 (9) PB1 (5) PB5 (7) PD1 (5) PD3 (1)	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
`U2Rx`	58 61	PB4 (4) PD0 (4)	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
`U2Tx`	8 62	PE4 (5) PD1 (4)	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.

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

14.3 Functional Description

Each Stellaris® UART performs the functions of parallel-to-serial and serial-to-parallel conversions. It is similar in functionality to a 16C550 UART, but is not register compatible.

The UART is configured for transmit and/or receive via the `TXE` and `RXE` bits of the **UART Control (UARTCTL)** register (see page 502). 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.

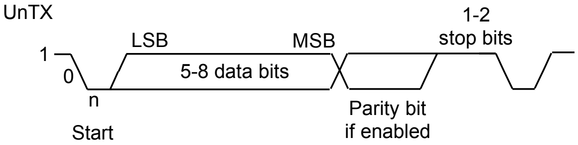
The UART module also includes a serial IR (SIR) encoder/decoder block that can be connected to an infrared transceiver to implement an IrDA SIR physical layer. The SIR function is programmed using the **UARTCTL** register.

14.3.1 Transmit/Receive Logic

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. The control logic outputs the serial bit stream beginning with a start bit and followed by the data bits (LSB first), parity bit, and the stop bits according to the programmed configuration in the control registers. See Figure 14-2 on page 481 for details.

The receive logic performs serial-to-parallel conversion on the received bit stream after a valid start pulse has been detected. Overrun, parity, frame error checking, and line-break detection are also performed, and their status accompanies the data that is written to the receive FIFO.

Figure 14-2. UART Character Frame



14.3.2 Baud-Rate Generation

The baud-rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. The number formed by these two values is used by the baud-rate generator to determine the bit period. Having a fractional baud-rate divider allows the UART to generate all the standard baud rates.

The 16-bit integer is loaded through the **UART Integer Baud-Rate Divisor (UARTIBRD)** register (see page 498) and the 6-bit fractional part is loaded with the **UART Fractional Baud-Rate Divisor (UARTFBRD)** register (see page 499). 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 = UARTSysClk / (ClkDiv * Baud Rate)
```

where `UARTSysClk` is the system clock connected to the UART, and `ClkDiv` is either 16 (if `HSE` in **UARTCTL** is clear) or 8 (if `HSE` is set).

The 6-bit fractional number (that is to be loaded into the `DIVFRAC` bit field in the **UARTFBRD** register) can be calculated by taking the fractional part of the baud-rate divisor, multiplying it by 64, and adding 0.5 to account for rounding errors:

```
UARTFBRD[DIVFRAC] = integer(BRDF * 64 + 0.5)
```

The UART generates an internal baud-rate reference clock at 8x or 16x the baud-rate (referred to as `Baud8` and `Baud16`, depending on the setting of the `HSE` bit (bit 5) in **UARTCTL**). This reference clock is divided by 8 or 16 to generate the transmit clock, and is used for error detection during receive operations.

Along with the **UART Line Control, High Byte (UARTLCRH)** register (see page 500), the **UARTIBRD** and **UARTFBRD** registers form an internal 30-bit register. This internal register is only updated when a write operation to **UARTLCRH** is performed, so any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register for the changes to take effect.

To update the baud-rate registers, there are four possible sequences:

- **UARTIBRD** write, **UARTFBRD** write, and **UARTLCRH** write
- **UARTFBRD** write, **UARTIBRD** write, and **UARTLCRH** write
- **UARTIBRD** write and **UARTLCRH** write
- **UARTFBRD** write and **UARTLCRH** write

14.3.3 Data Transmission

Data received or transmitted is stored in two 16-byte FIFOs, though the receive FIFO has an extra four bits per character for status information. For transmission, data is written into the transmit FIFO. If the UART is enabled, it causes a data frame to start transmitting with the parameters indicated in the **UARTLCRH** register. Data continues to be transmitted until there is no data left in the transmit

FIFO. The `BUSY` bit in the **UART Flag (UARTFR)** register (see page 495) is asserted as soon as data is written to the transmit FIFO (that is, if the FIFO is non-empty) and remains asserted while data is being transmitted. The `BUSY` bit is negated only when the transmit FIFO is empty, and the last character has been transmitted from the shift register, including the stop bits. The UART can indicate that it is busy even though the UART may no longer be enabled.

When the receiver is idle (the `UnRx` signal is continuously 1), and the data input goes Low (a start bit has been received), the receive counter begins running and data is sampled on the eighth cycle of `Baud16` or fourth cycle of `Baud8` depending on the setting of the `HSE` bit (bit 5) in **UARTCTL** (described in “Transmit/Receive Logic” on page 480).

The start bit is valid if the `UnRx` signal is still low on the eighth cycle of `Baud16` (`HSE` clear) or the fourth cycle of Baud 8 (`HSE` set), otherwise a false start bit is detected and is ignored. Start bit errors can be viewed in the **UART Receive Status (UARTRSR)** register (see page 492). If the start bit was valid, successive data bits are sampled on every 16th cycle of `Baud16` or 8th cycle of `Baud8` (that is, one bit period later) according to the programmed length of the data characters and value of the `HSE` bit in **UARTCTL**. The parity bit is then checked if parity mode is enabled. Data length and parity are defined in the **UARTLCRH** register.

Lastly, a valid stop bit is confirmed if the `UnRx` signal is High, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO along with any error bits associated with that word.

14.3.4 Serial IR (SIR)

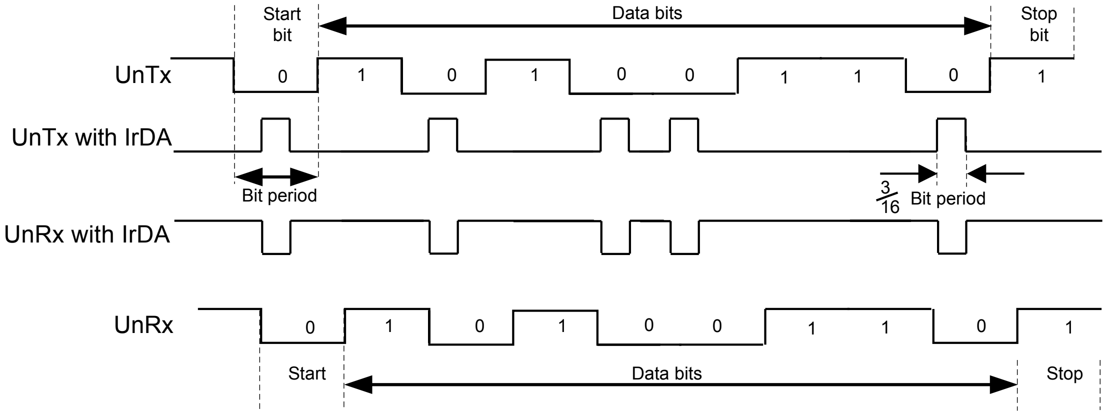
The UART peripheral includes an IrDA serial-IR (SIR) encoder/decoder block. The IrDA SIR block provides functionality that converts between an asynchronous UART data stream and a half-duplex serial SIR interface. No analog processing is performed on-chip. The role of the SIR block is to provide a digital encoded output and decoded input to the UART. When enabled, the SIR block uses the `UnTx` and `UnRx` pins for the SIR protocol. These signals should be connected to an infrared transceiver to implement an IrDA SIR physical layer link. The SIR block can receive and transmit, but it is only half-duplex so it cannot do both at the same time. Transmission must be stopped before data can be received. The IrDA SIR physical layer specifies a minimum 10-ms delay between transmission and reception.The SIR block has two modes of operation:

- In normal IrDA mode, a zero logic level is transmitted as a high pulse of 3/16th duration of the selected baud rate bit period on the output pin, while logic one levels are transmitted as a static LOW signal. These levels control the driver of an infrared transmitter, sending a pulse of light for each zero. On the reception side, the incoming light pulses energize the photo transistor base of the receiver, pulling its output LOW and driving the UART input pin LOW.

- In low-power IrDA mode, the width of the transmitted infrared pulse is set to three times the period of the internally generated IrLPBaud16 signal (1.63 µs, assuming a nominal 1.8432 MHz frequency) by changing the appropriate bit in the **UARTCR** register. See page 497 for more information on IrDA low-power pulse-duration configuration.

Figure 14-3 on page 483 shows the UART transmit and receive signals, with and without IrDA modulation.

Figure 14-3. IrDA Data Modulation



In both normal and low-power IrDA modes:

- During transmission, the UART data bit is used as the base for encoding
- During reception, the decoded bits are transferred to the UART receive logic

The IrDA SIR physical layer specifies a half-duplex communication link, with a minimum 10-ms delay between transmission and reception. This delay must be generated by software because it is not automatically supported by the UART. The delay is required because the infrared receiver electronics might become biased or even saturated from the optical power coupled from the adjacent transmitter LED. This delay is known as latency or receiver setup time.

14.3.5 ISO 7816 Support

The UART offers basic support to allow communication with an ISO 7816 smartcard. When bit 3 (`SMART`) of the **UARTCTL** register is set, the `UnTx` signal is used as a bit clock, and the `UnRx` signal is used as the half-duplex communication line connected to the smartcard. A GPIO signal can be used to generate the reset signal to the smartcard. The remaining smartcard signals should be provided by the system design.

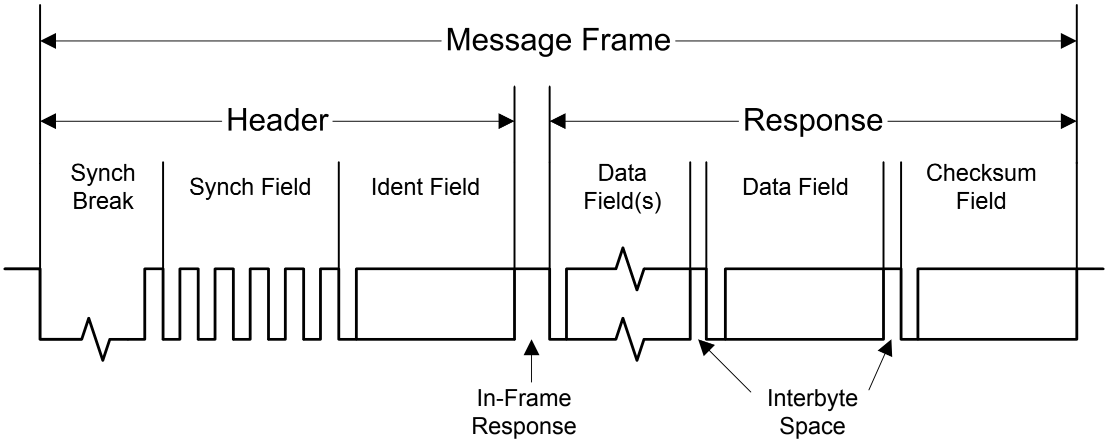
When using ISO 7816 mode, the **UARTLCRH** register must be set to transmit 8-bit words (`WLEN` bits 6:5 configured to 0x3) with EVEN parity (`PEN` set and `EPS` set). In this mode, the UART automatically uses 2 stop bits, and the `STP2` bit of the **UARTLCRH** register is ignored.

If a parity error is detected during transmission, UnRx is pulled Low during the second stop bit. In this case, the UART aborts the transmission, flushes the transmit FIFO and discards any data it contains, and raises a parity error interrupt, allowing software to detect the problem and initiate retransmission of the affected data. Note that the UART does not support automatic retransmission in this case.

14.3.6 LIN Support

The UART module offers hardware support for the LIN protocol as either a master or a slave. The LIN mode is enabled by setting the `LIN` bit in the **UARTCTL** register. A LIN message is identified by the use of a Sync Break at the beginning of the message. The Sync Break is a transmission of a series of 0s. The Sync Break is followed by the Sync data field (0x55). Figure 14-4 on page 484 illustrates the structure of a LIN message.

Figure 14-4. LIN Message



The UART should be configured as followed to operate in LIN mode:

1. Configure the UART for 1 start bit, 8 data bits, no parity, and 1 stop bit. Enable the Transmit FIFO.
2. Set the `LIN` bit in the **UARTCTL** register.

When preparing to send a LIN message, the TXFIFO should contain the Sync data (0x55) at FIFO location 0 and the Identifier data at location 1, followed by the data to be transmitted, and with the checksum in the final FIFO entry.

14.3.6.1 LIN Master

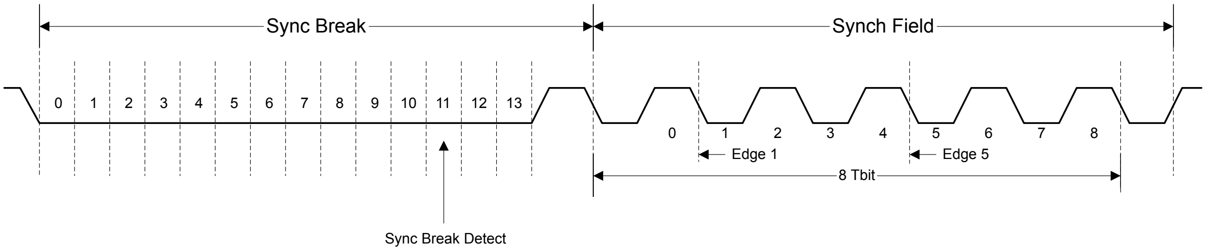
The UART is enabled to be the LIN master by setting the `MASTER` bit in the **UARTLCTL** register. The length of the Sync Break is programmable using the `BLEN` field in the **UARTLCTL** register and can be 13-16 bits (baud clock cycles).

14.3.6.2 LIN Slave

The LIN UART slave is required to adjust its baud rate to that of the LIN master. In slave mode, the LIN UART recognizes the Sync Break, which must be at least 13 bits in duration. A timer is provided to capture timing data on the 1st and 5th falling edges of the Sync field so that the baud rate can be adjusted to match the master.

After detecting a Sync Break, the UART waits for the synchronization field. The first falling edge generates an interrupt using the `LME1RIS` bit in the **UARTRIS** register, and the timer value is captured and stored in the **UARTLSS** register (T1). On the fifth falling edge, a second interrupt is generated using the `LME5RIS` bit in the **UARTRIS** register, and the timer value is captured again (T2). The actual baud rate can be calculated using (T2-T1)/8, and the local baud rate should be adjusted as needed. Figure 14-5 on page 485 illustrates the synchronization field.

Figure 14-5. LIN Synchronization Field



14.3.7 FIFO Operation

The UART has two 16-entry FIFOs; one for transmit and one for receive. Both FIFOs are accessed via the **UART Data (UARTDR)** register (see page 490). 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 500).

FIFO status can be monitored via the **UART Flag (UARTFR)** register (see page 495) 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 505). Both FIFOs can be individually configured to trigger interrupts at different levels. Available configurations include ⅛, ¼, ½, ¾, and ⅞. For example, if the ¼ 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 ½ mark.

14.3.8 Interrupts

The UART can generate interrupts when the following conditions are observed:

- Overrun Error
- Break Error
- Parity Error
- Framing Error
- Receive Timeout
- Transmit (when condition defined in the `TXIFLSEL` bit in the **UARTIFLS** register is met, or if the `EOT` bit in **UARTCTRL** is set, when the last bit of all transmitted data leaves the serializer)
- Receive (when condition defined in the `RXIFLSEL` bit in the **UARTIFLS** register is met)

All of the interrupt events are ORed together before being sent to the interrupt controller, so the UART can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine by reading the **UART Masked Interrupt Status (UARTMIS)** register (see page 514).

The interrupt events that can trigger a controller-level interrupt are defined in the **UART Interrupt Mask (UARTIM)** register (see page 507) by setting the corresponding `IM` bits. If interrupts are not used, the raw interrupt status is always visible via the **UART Raw Interrupt Status (UARTRIS)** register (see page 510).

Interrupts are always cleared (for both the **UARTMIS** and **UARTRIS** registers) by writing a 1 to the corresponding bit in the **UART Interrupt Clear (UARTICR)** register (see page 517).

The receive timeout interrupt is asserted when the receive FIFO is not empty, and no further data is received over a 32-bit period. The receive timeout interrupt is cleared either when the FIFO becomes empty through reading all the data (or by reading the holding register), or when a 1 is written to the corresponding bit in the **UARTICR** register.

14.3.9 Loopback Operation

The UART can be placed into an internal loopback mode for diagnostic or debug work by setting the `LBE` bit in the **UARTCTL** register (see page 502). In loopback mode, data transmitted on the `UnTx` output is received on the `UnRx` input.

14.3.10 DMA Operation

The UART provides an interface to the μDMA controller with separate channels for transmit and receive. The DMA operation of the UART is enabled through the **UART DMA Control (UARTDMACTL)** register. When DMA operation is enabled, the UART asserts a DMA request on the receive or transmit channel when the associated FIFO can transfer data. For the receive channel, a single transfer request is asserted whenever any data is in the receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is at or above the FIFO trigger level configured in the **UARTIFLS** register. For the transmit channel, a single transfer request is asserted whenever there is at least one empty location in the transmit FIFO. The burst request is asserted whenever the transmit FIFO contains fewer characters than the FIFO trigger level. The single and burst DMA transfer requests are handled automatically by the μDMA controller depending on how the DMA channel is configured.

To enable DMA operation for the receive channel, set the `RXDMAE` bit of the **DMA Control (UARTDMACTL)** register. To enable DMA operation for the transmit channel, set the `TXDMAE` bit of the **UARTDMACTL** register. The UART can also be configured to stop using DMA for the receive channel if a receive error occurs. If the `DMAERR` bit of the **UARTDMACR** register is set and a receive error occurs, the DMA receive requests are automatically disabled. This error condition can be cleared by clearing the appropriate UART error interrupt.

If DMA is enabled, then the μDMA controller triggers an interrupt when a transfer is complete. The interrupt occurs on the UART interrupt vector. Therefore, if interrupts are used for UART operation and DMA is enabled, the UART interrupt handler must be designed to handle the μDMA completion interrupt.

See “Micro Direct Memory Access (μDMA)” on page 227 for more details about programming the μDMA controller.

14.4 Initialization and Configuration

To enable and initialize the UART, the following steps are necessary:

1. The peripheral clock must be enabled by setting the `UART0`, `UART1`, or `UART2` bits in the **RCGC1** register (see page 147).

2. The clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module (see page 156).
3. Set the GPIO `AFSEL` bits for the appropriate pins (see page 307). To determine which GPIOs to configure, see Table 19-4 on page 636.
4. Configure the GPIO current level and/or slew rate as specified for the mode selected (see page 309 and page 317).
5. Configure the `PMCn` fields in the **GPIOPCTL** register to assign the UART signals to the appropriate pins (see page 324 and Table 19-5 on page 640).

To use the UARTs, the peripheral clock must be enabled by setting the `UART0`, `UART1`, or `UART2` bits in the **RCGC1** register (see page 147). In addition, the clock to the appropriate GPIO module must be enabled via the **RCGC2** register in the System Control module (see page 156). To find out which GPIO port to enable, refer to Table 19-5 on page 640.

This section discusses the steps that are required to use a UART module. For this example, the UART clock is assumed to be 20 MHz, and the desired UART configuration is:

- 115200 baud rate
- Data length of 8 bits
- One stop bit
- No parity
- FIFOs disabled
- No interrupts

The first thing to consider when programming the UART is the baud-rate divisor (BRD), because the **UARTIBRD** and **UARTFBRD** registers must be written before the **UARTLCRH** register. Using the equation described in “Baud-Rate Generation” on page 481, the BRD can be calculated:

```
BRD = 20,000,000 / (16 * 115,200) = 10.8507
```

which means that the `DIVINT` field of the **UARTIBRD** register (see page 498) should be set to 10 decimal or 0xA. The value to be loaded into the **UARTFBRD** register (see page 499) is calculated by the equation:

```
UARTFBRD[DIVFRAC] = integer(0.8507 * 64 + 0.5) = 54
```

With the BRD values in hand, the UART configuration is written to the module in the following order:

1. Disable the UART by clearing the `UARTEN` bit in the **UARTCTL** register.
2. Write the integer portion of the BRD to the **UARTIBRD** register.
3. Write the fractional portion of the BRD to the **UARTFBRD** register.
4. Write the desired serial parameters to the **UARTLCRH** register (in this case, a value of 0x0000.0060).
5. Optionally, configure the µDMA channel (see “Micro Direct Memory Access (µDMA)” on page 227) and enable the DMA option(s) in the **UARTDMACTL** register.

6. Enable the UART by setting the `UARTEN` bit in the **UARTCTL** register.

14.5 Register Map

Table 14-2 on page 488 lists the UART registers. The offset listed is a hexadecimal increment to the register's address, relative to that UART's base address:

- UART0: 0x4000.C000
- UART1: 0x4000.D000
- UART2: 0x4000.E000

Note that the UART module clock must be enabled before the registers can be programmed (see page 147).

Note: The UART must be disabled (see the `UARTEN` bit in the **UARTCTL** register on page 502) before any of the control registers are reprogrammed. When the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

Table 14-2. UART Register Map

Offset	Name	Type	Reset	Description	See page
0x000	UARTDR	R/W	0x0000.0000	UART Data	490
0x004	UARTRSR/UARTECR	R/W	0x0000.0000	UART Receive Status/Error Clear	492
0x018	UARTFR	RO	0x0000.0090	UART Flag	495
0x020	UARTILPR	R/W	0x0000.0000	UART IrDA Low-Power Register	497
0x024	UARTIBRD	R/W	0x0000.0000	UART Integer Baud-Rate Divisor	498
0x028	UARTFBRD	R/W	0x0000.0000	UART Fractional Baud-Rate Divisor	499
0x02C	UARTLCRH	R/W	0x0000.0000	UART Line Control	500
0x030	UARTCTL	R/W	0x0000.0300	UART Control	502
0x034	UARTIFLS	R/W	0x0000.0012	UART Interrupt FIFO Level Select	505
0x038	UARTIM	R/W	0x0000.0000	UART Interrupt Mask	507
0x03C	UARTRIS	RO	0x0000.000F	UART Raw Interrupt Status	510
0x040	UARTMIS	RO	0x0000.0000	UART Masked Interrupt Status	514
0x044	UARTICR	W1C	0x0000.0000	UART Interrupt Clear	517
0x048	UARTDMACTL	R/W	0x0000.0000	UART DMA Control	519
0x090	UARTLCTL	R/W	0x0000.0000	UART LIN Control	520
0x094	UARTLSS	RO	0x0000.0000	UART LIN Snap Shot	521
0x098	UARTLTIM	RO	0x0000.0000	UART LIN Timer	522
0xFD0	UARTPeriphID4	RO	0x0000.0000	UART Peripheral Identification 4	523
0xFD4	UARTPeriphID5	RO	0x0000.0000	UART Peripheral Identification 5	524
0xFD8	UARTPeriphID6	RO	0x0000.0000	UART Peripheral Identification 6	525
0xFDC	UARTPeriphID7	RO	0x0000.0000	UART Peripheral Identification 7	526

Table 14-2. UART Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xFE0	UARTPeriphID0	RO	0x0000.0060	UART Peripheral Identification 0	527
0xFE4	UARTPeriphID1	RO	0x0000.0000	UART Peripheral Identification 1	528
0xFE8	UARTPeriphID2	RO	0x0000.0018	UART Peripheral Identification 2	529
0xFEC	UARTPeriphID3	RO	0x0000.0001	UART Peripheral Identification 3	530
0xFF0	UARTPCellID0	RO	0x0000.000D	UART PrimeCell Identification 0	531
0xFF4	UARTPCellID1	RO	0x0000.00F0	UART PrimeCell Identification 1	532
0xFF8	UARTPCellID2	RO	0x0000.0005	UART PrimeCell Identification 2	533
0xFFC	UARTPCellID3	RO	0x0000.00B1	UART PrimeCell Identification 3	534

14.6 Register Descriptions

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

Register 1: UART Data (UARTDR), offset 0x000

Important: Use caution when reading this register. Performing a read may change bit status.

This register is the data register (the interface to the FIFOs).

For transmitted data, if the FIFO is enabled, data written to this location is pushed onto the transmit FIFO. If the FIFO is disabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). A write to this register initiates a transmission from the UART.

For received data, if the FIFO is enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO. If the FIFO is disabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO). The received data can be retrieved by reading this register.

UART Data (UARTDR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x000
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				OE	BE	PE	FE	DATA							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	OE	RO	0	UART Overrun Error Value Description 1 New data was received when the FIFO was full, resulting in data loss. 0 No data has been lost due to a FIFO overrun.
10	BE	RO	0	UART Break Error Value Description 1 A break condition has been detected, indicating that the receive data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits). 0 No break condition has occurred In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the received data input goes to a 1 (marking state), and the next valid start bit is received.

Bit/Field	Name	Type	Reset	Description
9	PE	RO	0	UART Parity Error Value Description 1 The parity of the received data character does not match the parity defined by bits 2 and 7 of the **UARTLCRH** register. 0 No parity error has occurred In FIFO mode, this error is associated with the character at the top of the FIFO.
8	FE	RO	0	UART Framing Error Value Description 1 The received character does not have a valid stop bit (a valid stop bit is 1). 0 No framing error has occurred
7:0	DATA	R/W	0x00	Data Transmitted or Received Data that is to be transmitted via the UART is written to this field. When read, this field contains the data that was received by the UART.

Register 2: UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004

The **UARTRSR/UARTECR** register is the receive status register/error clear register.

In addition to the **UARTDR** register, receive status can also be read from the **UARTRSR** register. If the status is read from this register, then the status information corresponds to the entry read from **UARTDR** prior to reading **UARTRSR**. The status information for overrun is set immediately when an overrun condition occurs.

The **UARTRSR** register cannot be written.

A write of any value to the **UARTECR** register clears the framing, parity, break, and overrun errors. All the bits are cleared on reset.

Read-Only Status Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x004
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OE	RO	0	UART Overrun Error Value Description 1 New data was received when the FIFO was full, resulting in data loss. 0 No data has been lost due to a FIFO overrun. This bit is cleared by a write to **UARTECR**. The FIFO contents remain valid because no further data is written when the FIFO is full, only the contents of the shift register are overwritten. The CPU must read the data in order to empty the FIFO.

Bit/Field	Name	Type	Reset	Description
2	BE	RO	0	UART Break Error Value Description 1 A break condition has been detected, indicating that the receive data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits). 0 No break condition has occurred This bit is cleared to 0 by a write to **UARTECR**. In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state) and the next valid start bit is received.
1	PE	RO	0	UART Parity Error Value Description 1 The parity of the received data character does not match the parity defined by bits 2 and 7 of the **UARTLCRH** register. 0 No parity error has occurred This bit is cleared to 0 by a write to **UARTECR**.
0	FE	RO	0	UART Framing Error Value Description 1 The received character does not have a valid stop bit (a valid stop bit is 1). 0 No framing error has occurred This bit is cleared to 0 by a write to **UARTECR**. In FIFO mode, this error is associated with the character at the top of the FIFO.

Write-Only Error Clear Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x004
Type WO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DATA							
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	WO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	WO	0x00	Error Clear A write to this register of any data clears the framing, parity, break, and overrun flags.

Register 3: UART Flag (UARTFR), offset 0x018

The **UARTFR** register is the flag register. After reset, the TXFF, RXFF, and BUSY bits are 0, and TXFE and RXFE bits are 1.

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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TXFE	RO	1	UART Transmit FIFO Empty The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. Value Description 1 If the FIFO is disabled (FEN is 0), the transmit holding register is empty. If the FIFO is enabled (FEN is 1), the transmit FIFO is empty. 0 The transmitter has data to transmit.
6	RXFF	RO	0	UART Receive FIFO Full The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. Value Description 1 If the FIFO is disabled (FEN is 0), the receive holding register is full. If the FIFO is enabled (FEN is 1), the receive FIFO is full. 0 The receiver can receive data.

Bit/Field	Name	Type	Reset	Description
5	TXFF	RO	0	UART Transmit FIFO Full The meaning of this bit depends on the state of the `FEN` bit in the **UARTLCRH** register. Value Description 1 If the FIFO is disabled (`FEN` is 0), the transmit holding register is full. If the FIFO is enabled (`FEN` is 1), the transmit FIFO is full. 0 The transmitter is not full.
4	RXFE	RO	1	UART Receive FIFO Empty The meaning of this bit depends on the state of the `FEN` bit in the **UARTLCRH** register. Value Description 1 If the FIFO is disabled (`FEN` is 0), the receive holding register is empty. If the FIFO is enabled (`FEN` is 1), the receive FIFO is empty. 0 The receiver is not empty.
3	BUSY	RO	0	UART Busy Value Description 1 The UART is busy transmitting data. This bit remains set until the complete byte, including all stop bits, has been sent from the shift register. 0 The UART is not busy. This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether UART is enabled).
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 4: UART IrDA Low-Power Register (UARTILPR), offset 0x020

The **UARTILPR** register stores the 8-bit low-power counter divisor value used to derive the low-power SIR pulse width clock by dividing down the system clock (SysClk). All the bits are cleared when reset.

The internal `IrLPBaud16` clock is generated by dividing down SysClk according to the low-power divisor value written to **UARTILPR**. The duration of SIR pulses generated when low-power mode is enabled is three times the period of the `IrLPBaud16` clock. The low-power divisor value is calculated as follows:

$$\mathtt{ILPDVSR = SysClk / F_{IrLPBaud16}}$$

where $F_{\mathtt{IrLPBaud16}}$ is nominally 1.8432 MHz.

The divisor must be programmed such that $1.42 \text{ MHz} < F_{\mathtt{IrLPBaud16}} < 2.12 \text{ MHz}$, resulting in a low-power pulse duration of 1.41–2.11 μs (three times the period of `IrLPBaud16`). The minimum frequency of `IrLPBaud16` ensures that pulses less than one period of `IrLPBaud16` are rejected, but pulses greater than 1.4 μs are accepted as valid pulses.

Note: Zero is an illegal value. Programming a zero value results in no `IrLPBaud16` pulses being generated.

UART IrDA Low-Power Register (UARTILPR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x020
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ILPDVSR	R/W	0x00	IrDA Low-Power Divisor This field contains the 8-bit low-power divisor value.

Register 5: UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024

The **UARTIBRD** register is the integer part of the baud-rate divisor value. All the bits are cleared on reset. The minimum possible divide ratio is 1 (when **UARTIBRD**=0), in which case the **UARTFBRD** register is ignored. When changing the **UARTIBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 481 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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DIVINT															
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	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 481 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	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	DIVFRAC	R/W	0x0	Fractional Baud-Rate Divisor

Register 7: UART Line Control (UARTLCRH), offset 0x02C

The **UARTLCRH** register is the line control register. Serial parameters such as data length, parity, and stop bit selection are implemented in this register.

When updating the baud-rate divisor (**UARTIBRD** and/or **UARTIFRD**), the **UARTLCRH** register must also be written. The write strobe for the baud-rate divisor registers is tied to the **UARTLCRH** register.

UART Line Control (UARTLCRH)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x02C
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	SPS	R/W	0	UART Stick Parity Select When bits 1, 2, and 7 of **UARTLCRH** are set, the parity bit is transmitted and checked as a 0. When bits 1 and 7 are set and 2 is cleared, the parity bit is transmitted and checked as a 1. When this bit is cleared, stick parity is disabled.
6:5	WLEN	R/W	0x0	UART Word Length The bits indicate the number of data bits transmitted or received in a frame as follows: Value Description 0x0 5 bits (default) 0x1 6 bits 0x2 7 bits 0x3 8 bits
4	FEN	R/W	0	UART Enable FIFOs Value Description 1 The transmit and receive FIFO buffers are enabled (FIFO mode). 0 The FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.

Bit/Field	Name	Type	Reset	Description
3	STP2	R/W	0	UART Two Stop Bits Select Value Description 1 Two stop bits are transmitted at the end of a frame. The receive logic does not check for two stop bits being received. When in 7816 smartcard mode (the `SMART` bit is set in the **UARTCTL** register), the number of stop bits is forced to 2. 0 One stop bit is transmitted at the end of a frame.
2	EPS	R/W	0	UART Even Parity Select Value Description 1 Even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits. 0 Odd parity is performed, which checks for an odd number of 1s. This bit has no effect when parity is disabled by the `PEN` bit.
1	PEN	R/W	0	UART Parity Enable Value Description 1 Parity checking and generation is enabled. 0 Parity is disabled and no parity bit is added to the data frame.
0	BRK	R/W	0	UART Send Break Value Description 1 A Low level is continually output on the `UnTx` signal, after completing transmission of the current character. For the proper execution of the break command, software must set this bit for at least two frames (character periods). 0 Normal use.

Register 8: UART Control (UARTCTL), offset 0x030

The **UARTCTL** register is the control register. All the bits are cleared on reset except for the Transmit Enable (`TXE`) and Receive Enable (`RXE`) bits, which are set.

To enable the UART module, the `UARTEN` bit must be set. If software requires a configuration change in the module, the `UARTEN` bit must be cleared before the configuration changes are written. If the UART is disabled during a transmit or receive operation, the current transaction is completed prior to the UART stopping.

Note: The **UARTCTL** register should not be changed while the UART is enabled or else the results are unpredictable. The following sequence is recommended for making changes to the **UARTCTL** register.

1. Disable the UART.
2. Wait for the end of transmission or reception of the current character.
3. Flush the transmit FIFO by clearing bit 4 (`FEN`) in the line control register (**UARTLCRH**).
4. Reprogram the control register.
5. Enable the UART.

UART Control (UARTCTL)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x030
Type R/W, reset 0x0000.0300

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved						RXE	TXE	LBE	LIN	HSE	EOT	SMART	SIRLP	SIREN	UARTEN
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	RXE	R/W	1	UART Receive Enable Value Description 1 The receive section of the UART is enabled. 0 The receive section of the UART is disabled. If the UART is disabled in the middle of a receive, it completes the current character before stopping. **Note:** To enable reception, the `UARTEN` bit must also be set.

Bit/Field	Name	Type	Reset	Description
8	TXE	R/W	1	UART Transmit Enable Value Description 1 The transmit section of the UART is enabled. 0 The transmit section of the UART is disabled. If the UART is disabled in the middle of a transmission, it completes the current character before stopping. **Note:** To enable transmission, the `UARTEN` bit must also be set.
7	LBE	R/W	0	UART Loop Back Enable Value Description 1 The `UnTx` path is fed through the `UnRx` path. 0 Normal operation.
6	LIN	R/W	0	LIN Mode Enable Value Description 1 The UART operates in LIN mode. 0 Normal operation.
5	HSE	R/W	0	High-Speed Enable Value Description 1 The UART is clocked using the system clock divided by 16. 0 The UART is clocked using the system clock divided by 8. **Note:** System clock used is also dependent on the baud-rate divisor configuration (see page 498) and page 499).
4	EOT	R/W	0	End of Transmission This bit determines the behavior of the `TXRIS` bit in the **UARTRIS** register. Value Description 1 The `TXRIS` bit is set only once all transmitted data, including stop bits, have cleared the serializer. 0 The `TXRIS` bit is set when the transmit FIFO condition specified in **UARTIFLS** is met.

Bit/Field	Name	Type	Reset	Description
3	SMART	R/W	0	ISO 7816 Smart Card Support Value Description 1 The UART operates in Smart Card mode. 0 Normal operation. The application must ensure that it sets 8-bit word length (`WLEN` set to `0x3`) and even parity (`PEN` set to 1, `EPS` set to 1, `SPS` set to 0) in **UARTLCRH** when using ISO 7816 mode. In this mode, the value of the `STP2` bit in **UARTLCRH** is ignored and the number of stop bits is forced to 2. Note that the UART does not support automatic retransmission on parity errors. If a parity error is detected on transmission, all further transmit operations are aborted and software must handle retransmission of the affected byte or message.
2	SIRLP	R/W	0	UART SIR Low-Power Mode This bit selects the IrDA encoding mode. Value Description 1 The UART operates in SIR Low-Power mode. Low-level bits are transmitted with a pulse width which is 3 times the period of the `IrLPBaud16` input signal, regardless of the selected bit rate. 0 Low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period. Setting this bit uses less power, but might reduce transmission distances. See page 497 for more information.
1	SIREN	R/W	0	UART SIR Enable Value Description 1 The IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol. 0 Normal operation.
0	UARTEN	R/W	0	UART Enable Value Description 1 The UART is enabled. 0 The UART is disabled. If 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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										RXIFLSEL			TXIFLSEL		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:3	RXIFLSEL	R/W	0x2	UART Receive Interrupt FIFO Level Select The trigger points for the receive interrupt are as follows:

Value	Description
0x0	RX FIFO ≥ ⅛ full
0x1	RX FIFO ≥ ¼ full
0x2	RX FIFO ≥ ½ full (default)
0x3	RX FIFO ≥ ¾ full
0x4	RX FIFO ≥ ⅞ full
0x5-0x7	Reserved

Bit/Field	Name	Type	Reset	Description
2:0	TXIFLSEL	R/W	0x2	UART Transmit Interrupt FIFO Level Select The trigger points for the transmit interrupt are as follows:

Value	Description
0x0	TX FIFO ≤ ⅛ full
0x1	TX FIFO ≤ ¼ full
0x2	TX FIFO ≤ ½ full (default)
0x3	TX FIFO ≤ ¾ full
0x4	TX FIFO ≤ ⅞ full
0x5-0x7	Reserved

Note: If the `EOT` bit in **UARTCTL** is set (see page 502), the transmit interrupt is generated once the FIFO is completely empty and all data including stop bits have left the transmit serializer. In this case, the setting of `TXIFLSEL` is ignored.

Register 10: UART Interrupt Mask (UARTIM), offset 0x038

The **UARTIM** register is the interrupt mask set/clear register.

On a read, this register gives the current value of the mask on the relevant interrupt. Setting a bit allows the corresponding raw interrupt signal to be routed to the interrupt controller. Clearing a bit prevents the raw interrupt signal from being sent to the interrupt controller.

UART Interrupt Mask (UARTIM)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x038
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LME5IM	LME1IM	LMSBIM	reserved		OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM	DSRIM	DCDIM	CTSIM	RIIM
Type	R/W	R/W	R/W	RO	RO	R/W	R/W	R/W	R/W	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	LME5IM	R/W	0	LIN Mode Edge 5 Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `LME5RIS` bit in the **UARTRIS** register is set. 0 The `LME5RIS` interrupt is suppressed and not sent to the interrupt controller.
14	LME1IM	R/W	0	LIN Mode Edge 1 Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `LME1RIS` bit in the **UARTRIS** register is set. 0 The `LME1RIS` interrupt is suppressed and not sent to the interrupt controller.
13	LMSBIM	R/W	0	LIN Mode Sync Break Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `LMSBRIS` bit in the **UARTRIS** register is set. 0 The `LMSBRIS` interrupt is suppressed and not sent to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
12:11	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEIM	R/W	0	UART Overrun Error Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `OERIS` bit in the **UARTRIS** register is set. 0 The `OERIS` interrupt is suppressed and not sent to the interrupt controller.
9	BEIM	R/W	0	UART Break Error Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `BERIS` bit in the **UARTRIS** register is set. 0 The `BERIS` interrupt is suppressed and not sent to the interrupt controller.
8	PEIM	R/W	0	UART Parity Error Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `PERIS` bit in the **UARTRIS** register is set. 0 The `PERIS` interrupt is suppressed and not sent to the interrupt controller.
7	FEIM	R/W	0	UART Framing Error Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `FERIS` bit in the **UARTRIS** register is set. 0 The `FERIS` interrupt is suppressed and not sent to the interrupt controller.
6	RTIM	R/W	0	UART Receive Time-Out Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `RTRIS` bit in the **UARTRIS** register is set. 0 The `RTRIS` interrupt is suppressed and not sent to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
5	TXIM	R/W	0	UART Transmit Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `TXRIS` bit in the **UARTRIS** register is set. 0 The `TXRIS` interrupt is suppressed and not sent to the interrupt controller.
4	RXIM	R/W	0	UART Receive Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `RXRIS` bit in the **UARTRIS** register is set. 0 The `RXRIS` interrupt is suppressed and not sent to the interrupt controller.
3	DSRIM	R/W	0	UART Data Set Ready Modem Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `DSRRIS` bit in the **UARTRIS** register is set. 0 The `DSRRIS` interrupt is suppressed and not sent to the interrupt controller.
2	DCDIM	R/W	0	UART Data Carrier Detect Modem Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `DCDRIS` bit in the **UARTRIS** register is set. 0 The `DCDRIS` interrupt is suppressed and not sent to the interrupt controller.
1	CTSIM	R/W	0	UART Clear to Send Modem Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `CTSRIS` bit in the **UARTRIS** register is set. 0 The `CTSRIS` interrupt is suppressed and not sent to the interrupt controller.
0	RIIM	R/W	0	UART Ring Indicator Modem Interrupt Mask Value Description 1 An interrupt is sent to the interrupt controller when the `RIRIS` bit in the **UARTRIS** register is set. 0 The `RIRIS` interrupt is suppressed and not sent to the interrupt controller.

Register 11: UART Raw Interrupt Status (UARTRIS), offset 0x03C

The **UARTRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt. A write has no effect.

UART Raw Interrupt Status (UARTRIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x03C
Type RO, reset 0x0000.000F

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LME5RIS	LME1RIS	LMSBRIS	reserved		OERIS	BERIS	PERIS	FERIS	RTRIS	TXRIS	RXRIS	DSRRIS	DCDRIS	CTSRIS	RIRIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	LME5RIS	RO	0	LIN Mode Edge 5 Raw Interrupt Status Value Description 1 The timer value at the 5th falling edge of the LIN Sync Field has been captured. 0 No interrupt This bit is cleared by writing a 1 to the `LME5IC` bit in the **UARTICR** register.
14	LME1RIS	RO	0	LIN Mode Edge 1 Raw Interrupt Status Value Description 1 The timer value at the 1st falling edge of the LIN Sync Field has been captured. 0 No interrupt This bit is cleared by writing a 1 to the `LME1IC` bit in the **UARTICR** register.
13	LMSBRIS	RO	0	LIN Mode Sync Break Raw Interrupt Status Value Description 1 A LIN Sync Break has been detected. 0 No interrupt This bit is cleared by writing a 1 to the `LMSBIC` bit in the **UARTICR** register.

Bit/Field	Name	Type	Reset	Description
12:11	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OERIS	RO	0	UART Overrun Error Raw Interrupt Status Value Description 1 An overrun error has occurred. 0 No interrupt This bit is cleared by writing a 1 to the `OEIC` bit in the **UARTICR** register.
9	BERIS	RO	0	UART Break Error Raw Interrupt Status Value Description 1 A break error has occurred. 0 No interrupt This bit is cleared by writing a 1 to the `BEIC` bit in the **UARTICR** register.
8	PERIS	RO	0	UART Parity Error Raw Interrupt Status Value Description 1 A parity error has occurred. 0 No interrupt This bit is cleared by writing a 1 to the `PEIC` bit in the **UARTICR** register.
7	FERIS	RO	0	UART Framing Error Raw Interrupt Status Value Description 1 A framing error has occurred. 0 No interrupt This bit is cleared by writing a 1 to the `FEIC` bit in the **UARTICR** register.
6	RTRIS	RO	0	UART Receive Time-Out Raw Interrupt Status Value Description 1 A receive time out has occurred. 0 No interrupt This bit is cleared by writing a 1 to the `RTIC` bit in the **UARTICR** register.

Bit/Field	Name	Type	Reset	Description
5	TXRIS	RO	0	UART Transmit Raw Interrupt Status Value Description 1 If the `EOT` bit in the **UARTCTRL** register is clear, the transmit FIFO level has passed through the condition defined in the **UARTIFLS** register. If the `EOT` bit is clear, the last bit of all transmitted data and flags has left the serializer. 0 No interrupt This bit is cleared by writing a 1 to the `TXIC` bit in the **UARTICR** register.
4	RXRIS	RO	0	UART Receive Raw Interrupt Status Value Description 1 The receive FIFO level has passed through the condition defined in the **UARTIFLS** register. 0 No interrupt This bit is cleared by writing a 1 to the `RXIC` bit in the **UARTICR** register.
3	DSRRIS	RO	0	UART Data Set Ready Modem Raw Interrupt Status Value Description 1 Data Set Ready used for software flow control. 0 No interrupt This bit is cleared by writing a 1 to the `DSRIC` bit in the **UARTICR** register.
2	DCDRIS	RO	0	UART Data Carrier Detect Modem Raw Interrupt Status Value Description 1 Data Carrier Detect used for software flow control. 0 No interrupt This bit is cleared by writing a 1 to the `DCDIC` bit in the **UARTICR** register.
1	CTSRIS	RO	0	UART Clear to Send Modem Raw Interrupt Status Value Description 1 Clear to Send used for software flow control. 0 No interrupt This bit is cleared by writing a 1 to the `CTSIC` bit in the **UARTICR** register.

Bit/Field	Name	Type	Reset	Description
0	RIRIS	RO	0	UART Ring Indicator Modem Raw Interrupt Status Value Description 1 Ring Indicator used for software flow control. 0 No interrupt This bit is cleared by writing a 1 to the `RIIC` bit in the **UARTICR** register.

Register 12: UART Masked Interrupt Status (UARTMIS), offset 0x040

The **UARTMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

UART Masked Interrupt Status (UARTMIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x040
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	LME5MIS	LME1MIS	LMSBMIS	reserved		OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS	DSRMIS	DCDMIS	CTSMIS	RIMIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	LME5MIS	RO	0	LIN Mode Edge 5 Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to the 5th falling edge of the LIN Sync Field. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `LME5IC` bit in the **UARTICR** register.
14	LME1MIS	RO	0	LIN Mode Edge 1 Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to the 1st falling edge of the LIN Sync Field. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `LME1IC` bit in the **UARTICR** register.
13	LMSBMIS	RO	0	LIN Mode Sync Break Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due the receipt of a LIN Sync Break. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `LMSBIC` bit in the **UARTICR** register.

Bit/Field	Name	Type	Reset	Description
12:11	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEMIS	RO	0	UART Overrun Error Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to an overrun error. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `OEIC` bit in the **UARTICR** register.
9	BEMIS	RO	0	UART Break Error Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a break error. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `BEIC` bit in the **UARTICR** register.
8	PEMIS	RO	0	UART Parity Error Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a parity error. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `PEIC` bit in the **UARTICR** register.
7	FEMIS	RO	0	UART Framing Error Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a framing error. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `FEIC` bit in the **UARTICR** register.
6	RTMIS	RO	0	UART Receive Time-Out Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to a receive time out. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `RTIC` bit in the **UARTICR** register.
5	TXMIS	RO	0	UART Transmit Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to passing through the specified transmit FIFO level (if the `EOT` bit is clear) or due to the transmission of the last data bit (if the `EOT` bit is set). 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `TXIC` bit in the **UARTICR** register.

Bit/Field	Name	Type	Reset	Description
4	RXMIS	RO	0	UART Receive Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to passing through the specified receive FIFO level. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `RXIC` bit in the **UARTICR** register.
3	DSRMIS	RO	0	UART Data Set Ready Modem Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to Data Set Ready. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `DSRIC` bit in the **UARTICR** register.
2	DCDMIS	RO	0	UART Data Carrier Detect Modem Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to Data Carrier Detect. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `DCDIC` bit in the **UARTICR** register.
1	CTSMIS	RO	0	UART Clear to Send Modem Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to Clear to Send. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `CTSIC` bit in the **UARTICR** register.
0	RIMIS	RO	0	UART Ring Indicator Modem Masked Interrupt Status Value Description 1 An unmasked interrupt was signaled due to Ring Indicator. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `RIIC` bit in the **UARTICR** register.

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
	LME5MIC	LME1MIC	LMSBMIC	reserved		OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC	DSRMIC	DCDMIC	CTSMIC	RIMIC
Type	W1C	W1C	W1C	RO	RO	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	LME5MIC	W1C	0	LIN Mode Edge 5 Interrupt Clear Writing a 1 to this bit clears the `LME5RIS` bit in the **UARTRIS** register and the `LME5MIS` bit in the **UARTMIS** register.
14	LME1MIC	W1C	0	LIN Mode Edge 1 Interrupt Clear Writing a 1 to this bit clears the `LME1RIS` bit in the **UARTRIS** register and the `LME1MIS` bit in the **UARTMIS** register.
13	LMSBMIC	W1C	0	LIN Mode Sync Break Interrupt Clear Writing a 1 to this bit clears the `LMSBRIS` bit in the **UARTRIS** register and the `LMSBMIS` bit in the **UARTMIS** register.
12:11	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEIC	W1C	0	Overrun Error Interrupt Clear Writing a 1 to this bit clears the `OERIS` bit in the **UARTRIS** register and the `OEMIS` bit in the **UARTMIS** register.
9	BEIC	W1C	0	Break Error Interrupt Clear Writing a 1 to this bit clears the `BERIS` bit in the **UARTRIS** register and the `BEMIS` bit in the **UARTMIS** register.
8	PEIC	W1C	0	Parity Error Interrupt Clear Writing a 1 to this bit clears the `PERIS` bit in the **UARTRIS** register and the `PEMIS` bit in the **UARTMIS** register.

Bit/Field	Name	Type	Reset	Description
7	FEIC	W1C	0	Framing Error Interrupt Clear Writing a 1 to this bit clears the `FERIS` bit in the **UARTRIS** register and the `FEMIS` bit in the **UARTMIS** register.
6	RTIC	W1C	0	Receive Time-Out Interrupt Clear Writing a 1 to this bit clears the `RTRIS` bit in the **UARTRIS** register and the `RTMIS` bit in the **UARTMIS** register.
5	TXIC	W1C	0	Transmit Interrupt Clear Writing a 1 to this bit clears the `TXRIS` bit in the **UARTRIS** register and the `TXMIS` bit in the **UARTMIS** register.
4	RXIC	W1C	0	Receive Interrupt Clear Writing a 1 to this bit clears the `RXRIS` bit in the **UARTRIS** register and the `RXMIS` bit in the **UARTMIS** register.
3	DSRMIC	W1C	0	UART Data Set Ready Modem Interrupt Clear Writing a 1 to this bit clears the `DSRRIS` bit in the **UARTRIS** register and the `DSRMIS` bit in the **UARTMIS** register.
2	DCDMIC	W1C	0	UART Data Carrier Detect Modem Interrupt Clear Writing a 1 to this bit clears the `DCDRIS` bit in the **UARTRIS** register and the `DCDMIS` bit in the **UARTMIS** register.
1	CTSMIC	W1C	0	UART Clear to Send Modem Interrupt Clear Writing a 1 to this bit clears the `CTSRIS` bit in the **UARTRIS** register and the `CTSMIS` bit in the **UARTMIS** register.
0	RIMIC	W1C	0	UART Ring Indicator Modem Interrupt Clear Writing a 1 to this bit clears the `RIRIS` bit in the **UARTRIS** register and the `RIMIS` bit in the **UARTMIS** register.

Register 14: UART DMA Control (UARTDMACTL), offset 0x048

The **UARTDMACTL** register is the DMA control register.

UART DMA Control (UARTDMACTL)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x048
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													DMAERR	TXDMAE	RXDMAE
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0x00000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	DMAERR	R/W	0	DMA on Error Value Description 1 µDMA receive requests are automatically disabled when a receive error occurs. 0 µDMA receive requests are unaffected when a receive error occurs.
1	TXDMAE	R/W	0	Transmit DMA Enable Value Description 1 µDMA for the transmit FIFO is enabled. 0 µDMA for the transmit FIFO is disabled.
0	RXDMAE	R/W	0	Receive DMA Enable Value Description 1 µDMA for the receive FIFO is enabled. 0 µDMA for the receive FIFO is disabled.

Register 15: UART LIN Control (UARTLCTL), offset 0x090

The **UARTLCTL** register is the configures the operation of the UART when in LIN mode.

UART LIN Control (UARTLCTL)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x090
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										BLEN		reserved			MASTER
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:4	BLEN	R/W	0x0	Sync Break Length Value Description 0x3 Sync break length is 16T bits 0x2 Sync break length is 15T bits 0x1 Sync break length is 14T bits 0x0 Sync break length is 13T bits (default)
3:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	MASTER	R/W	0	LIN Master Enable Value Description 1 The UART operates as a LIN master. 0 The UART operates as a LIN slave.

Register 16: UART LIN Snap Shot (UARTLSS), offset 0x094

The **UARTLSS** register captures the free-running timer value when either the Sync Edge 1 or the Sync Edge 5 is detected in LIN mode.

UART LIN Snap Shot (UARTLSS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x094
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TSS															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TSS	RO	0x0000	Timer Snap Shot This field contains the value of the free-running timer when either the Sync Edge 5 or the Sync Edge 1 was detected.

Register 17: UART LIN Timer (UARTLTIM), offset 0x098

The **UARTLTIM** register contains the current timer value for the free-running timer that is used to calculate the baud rate when in LIN slave mode. The value in this register is used along with the value in the **UART LIN Snap Shot (UARTLSS)** register to adjust the baud rate to match that of the master.

UART LIN Timer (UARTLTIM)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x098
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TIMER															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TIMER	RO	0x0000	Timer Value This field contains the value of the free-running timer.

Register 18: UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 4 (UARTPeriphID4)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFD0
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	UART Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

Register 19: UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 5 (UARTPeriphID5)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFD4
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	UART Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 20: UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 6 (UARTPeriphID6)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFD8
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID6							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	UART Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 21: UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 7 (UARTPeriphID7)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFDC
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	UART Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 22: UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 0 (UARTPeriphID0)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFE0
Type RO, reset 0x0000.0060

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x60	UART Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

Register 23: UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 1 (UARTPeriphID1)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFE4
Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	UART Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 24: UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 2 (UARTPeriphID2)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFE8
Type RO, reset 0x0000.0018

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	UART Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 25: UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 3 (UARTPeriphID3)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFEC
Type RO, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	UART Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 26: UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 0 (UARTPCellID0)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFF0
Type RO, reset 0x0000.000D

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	UART PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

Register 27: UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 1 (UARTPCellID1)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFF4
Type RO, reset 0x0000.00F0

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	UART PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

Register 28: UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 2 (UARTPCellID2)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFF8
Type RO, reset 0x0000.0005

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	UART PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

Register 29: UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 3 (UARTPCellID3)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0xFFC
Type RO, reset 0x0000.00B1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	UART PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

15 Synchronous Serial Interface (SSI)

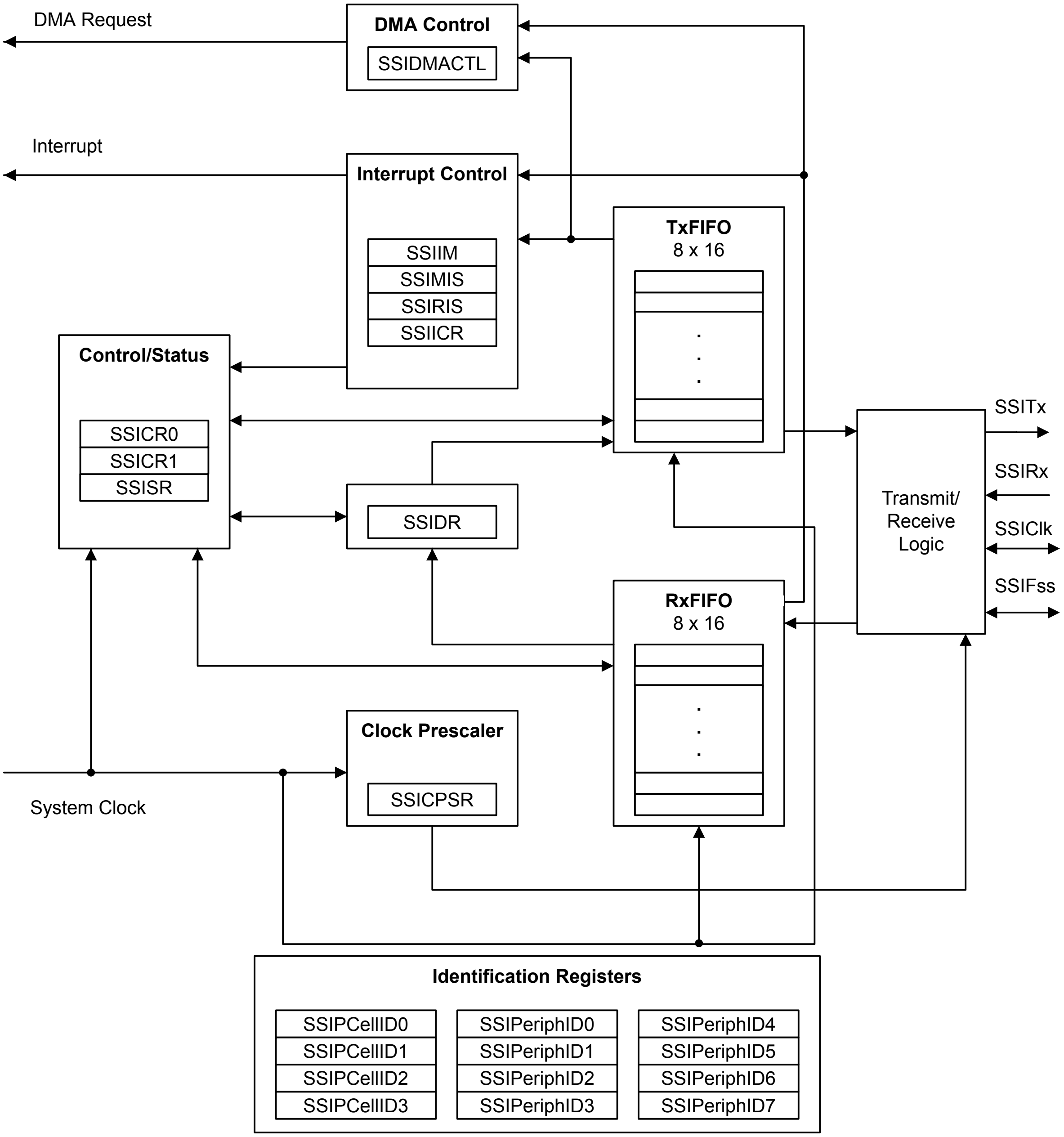
The Stellaris® microcontroller includes two Synchronous Serial Interface (SSI) modules. Each SSI is a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces.

The Stellaris® LM3S1W16 controller includes two SSI modules with the following features:

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

15.1 Block Diagram

Figure 15-1. SSI Module Block Diagram



15.2 Signal Description

Table 15-1 on page 537 lists the external signals of the SSI module and describes the function of each. The SSI signals are alternate functions for some GPIO signals and default to be GPIO signals at reset., with the exception of the `SSI0Clk`, `SSI0Fss`, `SSI0Rx`, and `SSI0Tx` pins which default to the SSI function. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the SSI signals. The `AFSEL` bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 307) should be set to choose the SSI function. The number in parentheses is the encoding that must be programmed into the `PMCn` field in the **GPIO Port Control (GPIOPCTL)** register (page 324) to assign the SSI signal to the specified GPIO port pin. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 285.

Table 15-1. Signals for SSI

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`SSI0Clk`	19	PA2 (1)	I/O	TTL	SSI module 0 clock.
`SSI0Fss`	20	PA3 (1)	I/O	TTL	SSI module 0 frame.
`SSI0Rx`	21	PA4 (1)	I	TTL	SSI module 0 receive.
`SSI0Tx`	22	PA5 (1)	O	TTL	SSI module 0 transmit.
`SSI1Clk`	6	PE0 (2)	I/O	TTL	SSI module 1 clock.
`SSI1Fss`	5	PE1 (2)	I/O	TTL	SSI module 1 frame.
`SSI1Rx`	2	PE2 (2)	I	TTL	SSI module 1 receive.
`SSI1Tx`	1	PE3 (2)	O	TTL	SSI module 1 transmit.

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

15.3 Functional Description

The SSI performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information. The transmit and receive paths are buffered with internal FIFO memories allowing up to eight 16-bit values to be stored independently in both transmit and receive modes. The SSI also supports the µDMA interface. The transmit and receive FIFOs can be programmed as destination/source addresses in the µDMA module. µDMA operation is enabled by setting the appropriate bit(s) in the **SSIDMACTL** register (see page 564).

15.3.1 Bit Rate Generation

The SSI includes a programmable bit rate clock divider and prescaler to generate the serial output clock. Bit rates are supported to 2 MHz and higher, although maximum bit rate is determined by peripheral devices.

The serial bit rate is derived by dividing down the input clock (SysClk). The clock is first divided by an even prescale value `CPSDVSR` from 2 to 254, which is programmed in the **SSI Clock Prescale (SSICPSR)** register (see page 557). 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 Control 0 (SSICR0)** register (see page 550).

The frequency of the output clock `SSIClk` is defined by:

```
SSIClk = SysClk / (CPSDVSR * (1 + SCR))
```

Note: For master mode, the system clock must be at least two times faster than the `SSIClk`. For slave mode, the system clock must be at least 12 times faster than the `SSIClk`.

See “Synchronous Serial Interface (SSI)” on page 655 to view SSI timing parameters.

15.3.2 FIFO Operation

15.3.2.1 Transmit FIFO

The common transmit FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. The CPU writes data to the FIFO by writing the **SSI Data (SSIDR)** register (see page 554), and data is stored in the FIFO until it is read out by the transmission logic.

When configured as a master or a slave, parallel data is written into the transmit FIFO prior to serial conversion and transmission to the attached slave or master, respectively, through the `SSITx` pin.

In slave mode, the SSI transmits data each time the master initiates a transaction. If the transmit FIFO is empty and the master initiates, the slave transmits the 8th most recent value in the transmit FIFO. If less than 8 values have been written to the transmit FIFO since the SSI module clock was enabled using the `SSI` bit in the **RGCG1** register, then 0 is transmitted. Care should be taken to ensure that valid data is in the FIFO as needed. The SSI can be configured to generate an interrupt or a µDMA request when the FIFO is empty.

15.3.2.2 Receive FIFO

The common receive FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. Received data from the serial interface is stored in the buffer until read out by the CPU, which accesses the read FIFO by reading the **SSIDR** register.

When configured as a master or slave, serial data received through the `SSIRx` pin is registered prior to parallel loading into the attached slave or master receive FIFO, respectively.

15.3.3 Interrupts

The SSI can generate interrupts when the following conditions are observed:

- Transmit FIFO service (when the transmit FIFO is half full or less)
- Receive FIFO service (when the receive FIFO is half full or more)
- Receive FIFO time-out
- Receive FIFO overrun
- End of transmission

All of the interrupt events are ORed together before being sent to the interrupt controller, so the SSI generates a single interrupt request to the controller regardless of the number of active interrupts. Each of the four individual maskable interrupts can be masked by clearing the appropriate bit in the **SSI Interrupt Mask (SSIIM)** register (see page 558). Setting the appropriate mask bit enables the interrupt.

The individual outputs, along with a combined interrupt output, allow use of either a global interrupt service routine or modular device drivers to handle interrupts. The transmit and receive dynamic dataflow interrupts have been separated from the status interrupts so that data can be read or written in response to the FIFO trigger levels. The status of the individual interrupt sources can be read from the **SSI Raw Interrupt Status (SSIRIS)** and **SSI Masked Interrupt Status (SSIMIS)** registers (see page 559 and page 561, respectively).

The receive FIFO has a time-out period that is 32 periods at the rate of `SSIClk` (whether or not `SSIClk` is currently active) and is started when the RX FIFO goes from EMPTY to not-EMPTY. If the RX FIFO is emptied before 32 clocks have passed, the time-out period is reset. As a result, the ISR should clear the Receive FIFO Time-out Interrupt just after reading out the RX FIFO by writing a 1 to the `RTIC` bit in the **SSI Interrupt Clear (SSIICR)** register. The interrupt should not be cleared so late that the ISR returns before the interrupt is actually cleared, or the ISR may be re-activated unnecessarily.

The End-of-Transmission (EOT) interrupt indicates that the data has been transmitted completely. This interrupt can be used to indicate when it is safe to turn off the SSI module clock or enter sleep mode. In addition, because transmitted data and received data complete at exactly the same time, the interrupt can also indicate that read data is ready immediately, without waiting for the receive FIFO time-out period to complete.

15.3.4 Frame Formats

Each data frame is between 4 and 16 bits long, depending on the size of data programmed, and is transmitted starting with the MSB. There are three basic frame types that can be selected:

- Texas Instruments synchronous serial
- Freescale SPI
- MICROWIRE

For all three formats, the serial clock (`SSIClk`) is held inactive while the SSI is idle, and `SSIClk` transitions at the programmed frequency only during active transmission or reception of data. The idle state of `SSIClk` is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Freescale SPI and MICROWIRE frame formats, the serial frame (`SSIFss`) pin is active Low, and is asserted (pulled down) during the entire transmission of the frame.

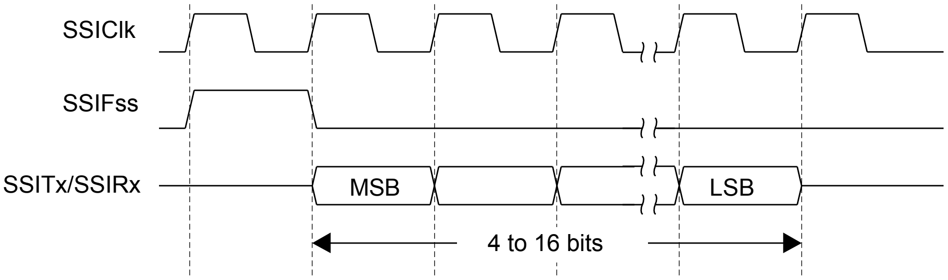
For Texas Instruments synchronous serial frame format, the `SSIFss` pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSI and the off-chip slave device drive their output data on the rising edge of `SSIClk` and latch data from the other device on the falling edge.

Unlike the full-duplex transmission of the other two frame formats, the MICROWIRE format uses a special master-slave messaging technique which operates at half-duplex. In this mode, when a frame begins, an 8-bit control message is transmitted to the off-chip slave. During this transmit, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the requested data. The returned data can be 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

15.3.4.1 Texas Instruments Synchronous Serial Frame Format

Figure 15-2 on page 539 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

Figure 15-2. TI Synchronous Serial Frame Format (Single Transfer)

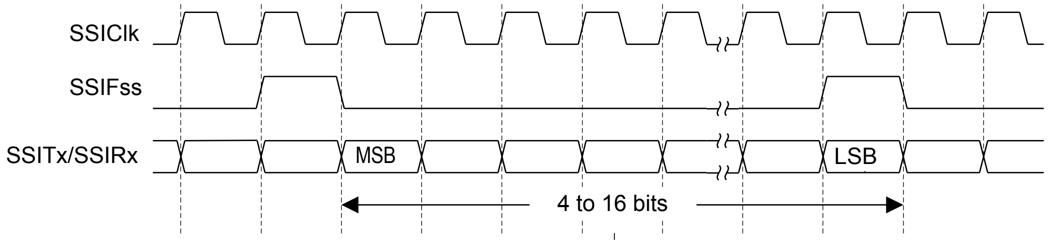


In this mode, `SSIClk` and `SSIFss` are forced Low, and the transmit data line `SSITx` is tristated whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, `SSIFss` is pulsed High for one `SSIClk` period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of `SSIClk`, the MSB of the 4 to 16-bit data frame is shifted out on the `SSITx` pin. Likewise, the MSB of the received data is shifted onto the `SSIRx` pin by the off-chip serial slave device.

Both the SSI and the off-chip serial slave device then clock each data bit into their serial shifter on each falling edge of `SSIClk`. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of `SSIClk` after the LSB has been latched.

Figure 15-3 on page 540 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

Figure 15-3. TI Synchronous Serial Frame Format (Continuous Transfer)



15.3.4.2 Freescale SPI Frame Format

The Freescale SPI interface is a four-wire interface where the `SSIFss` signal behaves as a slave select. The main feature of the Freescale SPI format is that the inactive state and phase of the `SSIClk` signal are programmable through the `SPO` and `SPH` bits in the **SSISCR0** control register.

SPO Clock Polarity Bit

When the `SPO` clock polarity control bit is clear, it produces a steady state Low value on the `SSIClk` pin. If the `SPO` bit is set, a steady state High value is placed on the `SSIClk` pin when data is not being transferred.

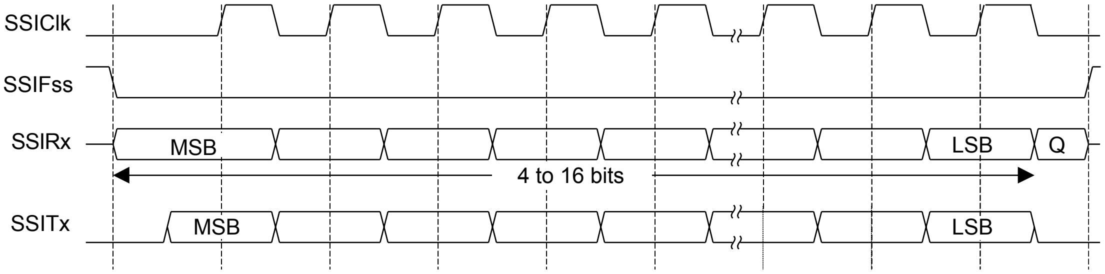
SPH Phase Control Bit

The `SPH` phase control bit selects the clock edge that captures data and allows it to change state. The state of this bit has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the `SPH` phase control bit is clear, data is captured on the first clock edge transition. If the `SPH` bit is set, data is captured on the second clock edge transition.

15.3.4.3 Freescale SPI Frame Format with SPO=0 and SPH=0

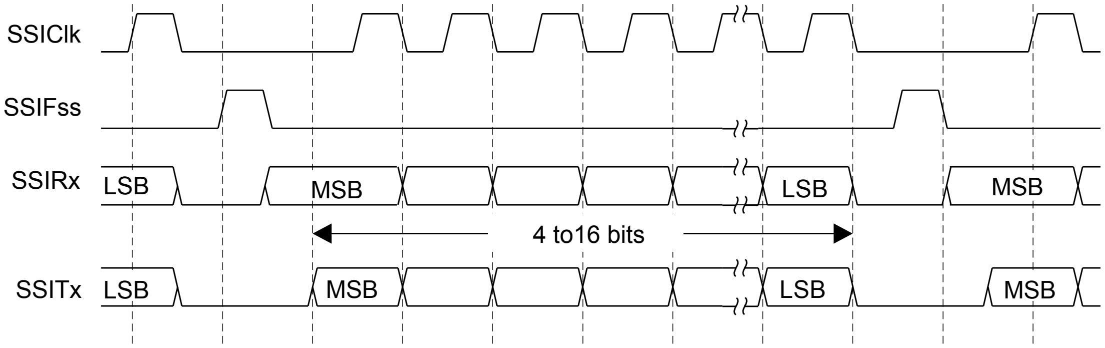
Single and continuous transmission signal sequences for Freescale SPI format with `SPO`=0 and `SPH`=0 are shown in Figure 15-4 on page 541 and Figure 15-5 on page 541.

Figure 15-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0



Note: Q is undefined.

Figure 15-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0



In this configuration, during idle periods:

- `SSIClk` is forced Low
- `SSIFss` is forced High
- The transmit data line `SSITx` is arbitrarily forced Low
- When the SSI is configured as a master, it enables the `SSIClk` pad
- When the SSI is configured as a slave, it disables the `SSIClk` pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the `SSIFss` master signal being driven Low, causing slave data to be enabled onto the `SSIRx` input line of the master. The master `SSITx` output pad is enabled.

One half `SSIClk` period later, valid master data is transferred to the `SSITx` pin. Once both the master and slave data have been set, the `SSIClk` master clock pin goes High after one additional half `SSIClk` period.

The data is now captured on the rising and propagated on the falling edges of the `SSIClk` signal.

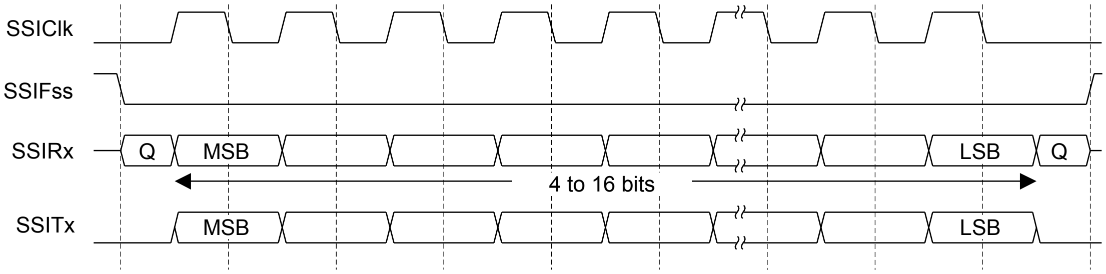
In the case of a single word transmission, after all bits of the data word have been transferred, the `SSIFss` line is returned to its idle High state one `SSIClk` period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the `SSIFss` signal must be pulsed High between each data word transfer because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the `SPH` bit is clear. Therefore, the master device must raise the `SSIFss` pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the `SSIFss` pin is returned to its idle state one `SSIClk` period after the last bit has been captured.

15.3.4.4 Freescale SPI Frame Format with SPO=0 and SPH=1

The transfer signal sequence for Freescale SPI format with `SPO`=0 and `SPH`=1 is shown in Figure 15-6 on page 542, which covers both single and continuous transfers.

Figure 15-6. Freescale SPI Frame Format with SPO=0 and SPH=1



Note: Q is undefined.

In this configuration, during idle periods:

- `SSIClk` is forced Low
- `SSIFss` is forced High
- The transmit data line `SSITx` is arbitrarily forced Low
- When the SSI is configured as a master, it enables the `SSIClk` pad
- When the SSI is configured as a slave, it disables the `SSIClk` pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the `SSIFss` master signal being driven Low. The master `SSITx` output is enabled. After an additional one-half `SSIClk` period, both master and slave valid data are enabled onto their respective transmission lines. At the same time, the `SSIClk` is enabled with a rising edge transition.

Data is then captured on the falling edges and propagated on the rising edges of the `SSIClk` signal.

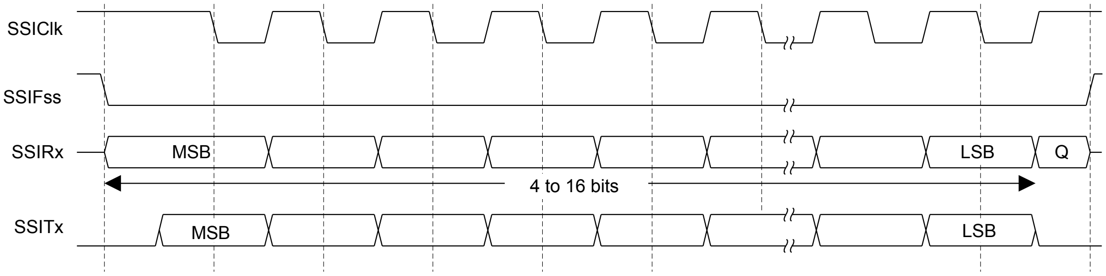
In the case of a single word transfer, after all bits have been transferred, the `SSIFss` line is returned to its idle High state one `SSIClk` period after the last bit has been captured.

For continuous back-to-back transfers, the `SSIFss` pin is held Low between successive data words, and termination is the same as that of the single word transfer.

15.3.4.5 Freescale SPI Frame Format with SPO=1 and SPH=0

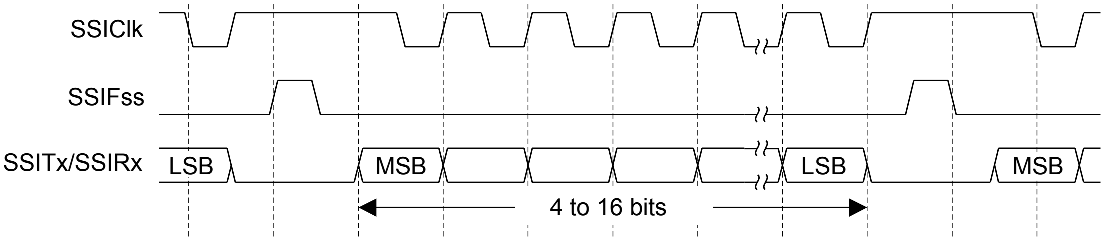
Single and continuous transmission signal sequences for Freescale SPI format with `SPO`=1 and `SPH`=0 are shown in Figure 15-7 on page 543 and Figure 15-8 on page 543.

Figure 15-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0



Note: Q is undefined.

Figure 15-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0



In this configuration, during idle periods:

- `SSIClk` is forced High
- `SSIFss` is forced High
- The transmit data line `SSITx` is arbitrarily forced Low
- When the SSI is configured as a master, it enables the `SSIClk` pad
- When the SSI is configured as a slave, it disables the `SSIClk` pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the `SSIFss` master signal being driven Low, causing slave data to be immediately transferred onto the `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. Once both the master and slave data have been set, the `SSIClk` master clock pin becomes Low after one additional half `SSIClk` period, meaning that data is captured on the falling edges and propagated on the rising edges of the `SSIClk` signal.

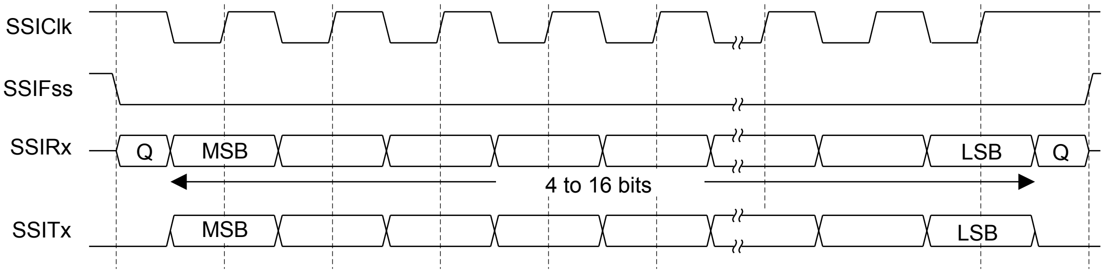
In the case of a single word transmission, after all bits of the data word are transferred, the `SSIFss` line is returned to its idle High state one `SSIClk` period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the `SSIFss` signal must be pulsed High between each data word transfer because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the `SPH` bit is clear. Therefore, the master device must raise the `SSIFss` pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the `SSIFss` pin is returned to its idle state one `SSIClk` period after the last bit has been captured.

15.3.4.6 Freescale SPI Frame Format with SPO=1 and SPH=1

The transfer signal sequence for Freescale SPI format with `SPO`=1 and `SPH`=1 is shown in Figure 15-9 on page 544, which covers both single and continuous transfers.

Figure 15-9. Freescale SPI Frame Format with SPO=1 and SPH=1



Note: Q is undefined.

In this configuration, during idle periods:

- `SSIClk` is forced High
- `SSIFss` is forced High
- The transmit data line `SSITx` is arbitrarily forced Low
- When the SSI is configured as a master, it enables the `SSIClk` pad
- When the SSI is configured as a slave, it disables the `SSIClk` pad

If the SSI is enabled and valid data is in the transmit FIFO, the start of transmission is signified by the `SSIFss` master signal being driven Low. The master `SSITx` output pad is enabled. After an additional one-half `SSIClk` period, both master and slave data are enabled onto their respective transmission lines. At the same time, `SSIClk` is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the `SSIClk` signal.

After all bits have been transferred, in the case of a single word transmission, the `SSIFss` line is returned to its idle high state one `SSIClk` period after the last bit has been captured.

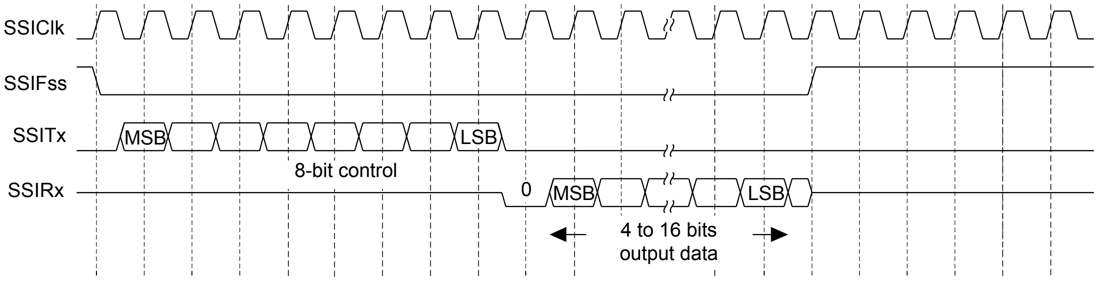
For continuous back-to-back transmissions, the `SSIFss` pin remains in its active Low state until the final bit of the last word has been captured and then returns to its idle state as described above.

For continuous back-to-back transfers, the `SSIFss` pin is held Low between successive data words and termination is the same as that of the single word transfer.

15.3.4.7 MICROWIRE Frame Format

Figure 15-10 on page 545 shows the MICROWIRE frame format for a single frame. Figure 15-11 on page 546 shows the same format when back-to-back frames are transmitted.

Figure 15-10. MICROWIRE Frame Format (Single Frame)



MICROWIRE format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex and uses a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSI to the off-chip slave device. During this transmission, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

In this configuration, during idle periods:

- `SSIClk` is forced Low
- `SSIFss` is forced High
- The transmit data line `SSITx` is arbitrarily forced Low

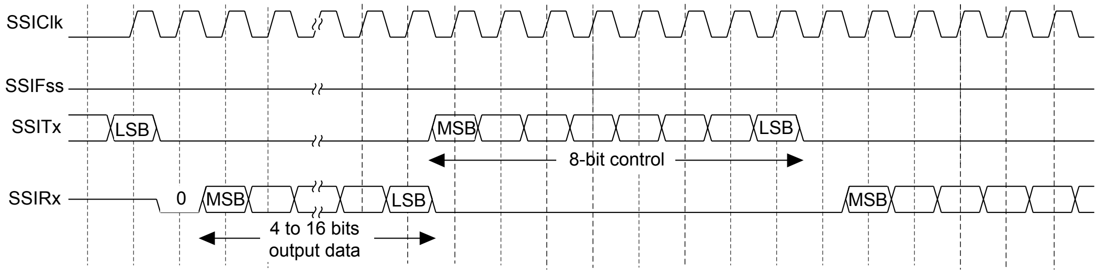
A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of `SSIFss` causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic and the MSB of the 8-bit control frame to be shifted out onto the `SSITx` pin. `SSIFss` remains Low for the duration of the frame transmission. The `SSIRx` pin remains tristated during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on each rising edge of `SSIClk`. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state, and the slave responds by transmitting data back to the SSI. Each bit is driven onto the `SSIRx` line on the falling edge of `SSIClk`. The SSI in turn latches each bit on the rising edge of `SSIClk`. At the end of the frame, for single transfers, the `SSIFss` signal is pulled High one clock period after the last bit has been latched in the receive serial shifter, causing the data to be transferred to the receive FIFO.

Note: The off-chip slave device can tristate the receive line either on the falling edge of `SSIClk` after the LSB has been latched by the receive shifter or when the `SSIFss` pin goes High.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the `SSIFss` line is continuously asserted (held Low) and transmission of data occurs back-to-back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge of `SSIClk`, after the LSB of the frame has been latched into the SSI.

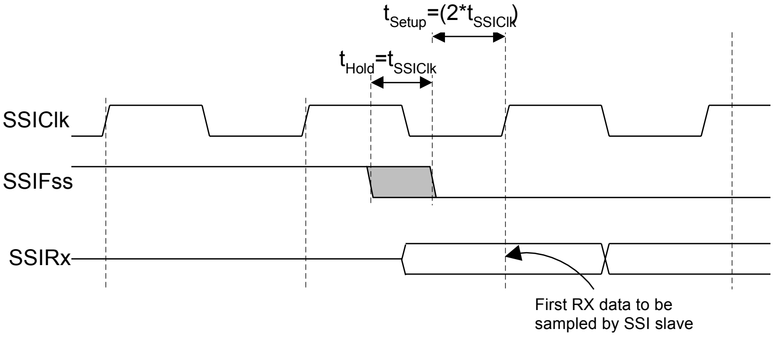
Figure 15-11. MICROWIRE Frame Format (Continuous Transfer)



In the MICROWIRE mode, the SSI slave samples the first bit of receive data on the rising edge of `SSIClk` after `SSIFss` has gone Low. Masters that drive a free-running `SSIClk` must ensure that the `SSIFss` signal has sufficient setup and hold margins with respect to the rising edge of `SSIClk`.

Figure 15-12 on page 546 illustrates these setup and hold time requirements. With respect to the `SSIClk` rising edge on which the first bit of receive data is to be sampled by the SSI slave, `SSIFss` must have a setup of at least two times the period of `SSIClk` on which the SSI operates. With respect to the `SSIClk` rising edge previous to this edge, `SSIFss` must have a hold of at least one `SSIClk` period.

Figure 15-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements



15.3.5 DMA Operation

The SSI peripheral provides an interface to the μDMA controller with separate channels for transmit and receive. The μDMA operation of the SSI is enabled through the **SSI DMA Control (SSIDMACTL)** register. When μDMA operation is enabled, the SSI asserts a μDMA request on the receive or transmit channel when the associated FIFO can transfer data. For the receive channel, a single transfer request is asserted whenever any data is in the receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is 4 or more items. For the transmit channel, a single transfer request is asserted whenever at least one empty location is in the transmit FIFO. The burst request is asserted whenever the transmit FIFO has 4 or more empty slots. The single and burst μDMA transfer requests are handled automatically by the μDMA controller depending how the μDMA channel is configured. To enable μDMA operation for the receive channel, the `RXDMAE` bit of the **DMA Control (SSIDMACTL)** register should be set. To enable μDMA operation for the transmit channel, the `TXDMAE` bit of **SSIDMACTL** should be set. If μDMA is enabled, then the μDMA controller triggers an interrupt when a transfer is complete. The interrupt occurs on the SSI interrupt vector. Therefore, if interrupts are used for SSI operation and μDMA is enabled, the SSI interrupt handler must be designed to handle the μDMA completion interrupt.

See “Micro Direct Memory Access (μDMA)” on page 227 for more details about programming the μDMA controller.

15.4 Initialization and Configuration

To enable and initialize the SSI, the following steps are necessary:

1. Enable the SSI module by setting the `SSI` bit in the **RCGC1** register (see page 147).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register (see page 156). To find out which GPIO port to enable, refer to Table 19-5 on page 640.
3. Set the GPIO `AFSEL` bits for the appropriate pins (see page 307). To determine which GPIOs to configure, see Table 19-4 on page 636.
4. Configure the `PMCn` fields in the **GPIOPCTL** register to assign the SSI signals to the appropriate pins. See page 324 and Table 19-5 on page 640.

For each of the frame formats, the SSI is configured using the following steps:

1. Ensure that the `SSE` bit in the **SSICR1** register is clear before making any configuration changes.
2. Select whether the SSI is a master or slave:
   a. For master operations, set the **SSICR1** register to 0x0000.0000.
   b. For slave mode (output enabled), set the **SSICR1** register to 0x0000.0004.
   c. For slave mode (output disabled), set the **SSICR1** register to 0x0000.000C.
3. Configure the clock prescale divisor by writing the **SSICPSR** register.
4. Write the **SSICR0** register with the following configuration:
   - Serial clock rate (`SCR`)
   - Desired clock phase/polarity, if using Freescale SPI mode (`SPH` and `SPO`)
   - The protocol mode: Freescale SPI, TI SSF, MICROWIRE (`FRF`)
   - The data size (`DSS`)
5. Optionally, configure the μDMA channel (see “Micro Direct Memory Access (μDMA)” on page 227) and enable the DMA option(s) in the **SSIDMACTL** register.
6. Enable the SSI by setting the `SSE` bit in the **SSICR1** register.

As an example, assume the SSI must be configured to operate with the following parameters:

- Master operation
- Freescale SPI mode (SPO=1, SPH=1)
- 1 Mbps bit rate
- 8 data bits

Assuming the system clock is 20 MHz, the bit rate calculation would be:

`SSIClk = SysClk / (CPSDVSR * (1 + SCR)) 1x106 = 20x106 / (CPSDVSR * (1 + SCR))`

In this case, if `CPSDVSR`=0x2, `SCR` must be 0x9.

The configuration sequence would be as follows:

1. Ensure that the `SSE` bit in the **SSICR1** register is clear.
2. Write the **SSICR1** register with a value of 0x0000.0000.
3. Write the **SSICPSR** register with a value of 0x0000.0002.
4. Write the **SSICR0** register with a value of 0x0000.09C7.
5. The SSI is then enabled by setting the `SSE` bit in the **SSICR1** register.

15.5 Register Map

Table 15-2 on page 548 lists the SSI registers. The offset listed is a hexadecimal increment to the register's address, relative to that SSI module's base address:

- SSI0: 0x4000.8000
- SSI1: 0x4000.9000

Note that the SSI module clock must be enabled before the registers can be programmed (see page 147).

Note: The SSI must be disabled (see the `SSE` bit in the **SSICR1** register) before any of the control registers are reprogrammed.

Table 15-2. SSI Register Map

Offset	Name	Type	Reset	Description	See page
0x000	SSICR0	R/W	0x0000.0000	SSI Control 0	550
0x004	SSICR1	R/W	0x0000.0000	SSI Control 1	552
0x008	SSIDR	R/W	0x0000.0000	SSI Data	554
0x00C	SSISR	RO	0x0000.0003	SSI Status	555
0x010	SSICPSR	R/W	0x0000.0000	SSI Clock Prescale	557
0x014	SSIIM	R/W	0x0000.0000	SSI Interrupt Mask	558
0x018	SSIRIS	RO	0x0000.0008	SSI Raw Interrupt Status	559
0x01C	SSIMIS	RO	0x0000.0000	SSI Masked Interrupt Status	561
0x020	SSIICR	W1C	0x0000.0000	SSI Interrupt Clear	563
0x024	SSIDMACTL	R/W	0x0000.0000	SSI DMA Control	564
0xFD0	SSIPeriphID4	RO	0x0000.0000	SSI Peripheral Identification 4	565
0xFD4	SSIPeriphID5	RO	0x0000.0000	SSI Peripheral Identification 5	566

Table 15-2. SSI Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
0xFD8	SSIPeriphID6	RO	0x0000.0000	SSI Peripheral Identification 6	567
0xFDC	SSIPeriphID7	RO	0x0000.0000	SSI Peripheral Identification 7	568
0xFE0	SSIPeriphID0	RO	0x0000.0022	SSI Peripheral Identification 0	569
0xFE4	SSIPeriphID1	RO	0x0000.0000	SSI Peripheral Identification 1	570
0xFE8	SSIPeriphID2	RO	0x0000.0018	SSI Peripheral Identification 2	571
0xFEC	SSIPeriphID3	RO	0x0000.0001	SSI Peripheral Identification 3	572
0xFF0	SSIPCellID0	RO	0x0000.000D	SSI PrimeCell Identification 0	573
0xFF4	SSIPCellID1	RO	0x0000.00F0	SSI PrimeCell Identification 1	574
0xFF8	SSIPCellID2	RO	0x0000.0005	SSI PrimeCell Identification 2	575
0xFFC	SSIPCellID3	RO	0x0000.00B1	SSI PrimeCell Identification 3	576

15.6 Register Descriptions

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

Register 1: SSI Control 0 (SSICR0), offset 0x000

The **SSICR0** register contains bit fields that control various functions within the SSI module. Functionality such as protocol mode, clock rate, and data size are configured in this register.

SSI Control 0 (SSICR0)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x000
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	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	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	SCR	R/W	0x00	SSI Serial Clock Rate This bit field is used to generate the transmit and receive bit rate of the SSI. The bit rate is: `BR=SSIClk/(CPSDVSR * (1 + SCR))` where `CPSDVSR` is an even value from 2-254 programmed in the **SSICPSR** register, and `SCR` is a value from 0-255.
7	SPH	R/W	0	SSI Serial Clock Phase This bit is only applicable to the Freescale SPI Format. The `SPH` control bit selects the clock edge that captures data and allows it to change state. This bit has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. Value Description 0 Data is captured on the first clock edge transition. 1 Data is captured on the second clock edge transition.
6	SPO	R/W	0	SSI Serial Clock Polarity Value Description 0 A steady state Low value is placed on the `SSIClk` pin. 1 A steady state High value is placed on the `SSIClk` pin when data is not being transferred.

Bit/Field	Name	Type	Reset	Description
5:4	FRF	R/W	0x0	SSI Frame Format Select Value Frame Format 0x0 Freescale SPI Frame Format 0x1 Texas Instruments Synchronous Serial Frame Format 0x2 MICROWIRE Frame Format 0x3 Reserved
3:0	DSS	R/W	0x0	SSI Data Size Select Value Data Size 0x0-0x2 Reserved 0x3 4-bit data 0x4 5-bit data 0x5 6-bit data 0x6 7-bit data 0x7 8-bit data 0x8 9-bit data 0x9 10-bit data 0xA 11-bit data 0xB 12-bit data 0xC 13-bit data 0xD 14-bit data 0xE 15-bit data 0xF 16-bit data

Register 2: SSI Control 1 (SSICR1), offset 0x004

The **SSICR1** register contains bit fields that control various functions within the SSI module. Master and slave mode functionality is controlled by this register.

SSI Control 1 (SSICR1)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x004
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											EOT	SOD	MS	SSE	LBM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	EOT	R/W	0	End of Transmission Value Description 0 The TXRIS interrupt indicates that the transmit FIFO is half full or less. 1 The End of Transmit interrupt mode for the TXRIS interrupt is enabled.
3	SOD	R/W	0	SSI Slave Mode Output Disable 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. Value Description 0 SSI can drive the `SSITx` output in Slave 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 the SSI is disabled (`SSE`=0). Value Description 0 The SSI is configured as a master. 1 The SSI is configured as a slave.

Bit/Field	Name	Type	Reset	Description
1	SSE	R/W	0	SSI Synchronous Serial Port Enable Value Description 0 SSI operation is disabled. 1 SSI operation is enabled. **Note:** This bit must be cleared before any control registers are reprogrammed.
0	LBM	R/W	0	SSI Loopback Mode Value Description 0 Normal serial port operation enabled. 1 Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.

Register 3: SSI Data (SSIDR), offset 0x008

Important: Use caution when reading this register. Performing a read may change bit status.

The **SSIDR** register is 16-bits wide. When the **SSIDR** register is read, the entry in the receive FIFO that is pointed to by the current FIFO read pointer is accessed. When a data value is removed by the SSI receive logic from the incoming data frame, it is placed into the entry in the receive FIFO pointed to by the current FIFO write pointer.

When the **SSIDR** register is written to, the entry in the transmit FIFO that is pointed to by the write pointer is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. Each data value is loaded into the transmit serial shifter, then serially shifted out onto the `SSITx` pin at the programmed bit rate.

When a data size of less than 16 bits is selected, the user must right-justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 bits is automatically right-justified in the receive buffer.

When the SSI is programmed for MICROWIRE frame format, the default size for transmit data is eight bits (the most significant byte is ignored). The receive data size is controlled by the programmer. The transmit FIFO and the receive FIFO are not cleared even when the `SSE` bit in the **SSICR1** register is cleared, allowing the software to fill the transmit FIFO before enabling the SSI.

SSI Data (SSIDR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x008
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DATA	R/W	0x0000	SSI Receive/Transmit Data A read operation reads the receive FIFO. A write operation writes the transmit FIFO. Software must right-justify data when the SSI is programmed for a data size that is less than 16 bits. Unused bits at the top are ignored by the transmit logic. The receive logic automatically right-justifies the data.

Register 4: SSI Status (SSISR), offset 0x00C

The **SSISR** register contains bits that indicate the FIFO fill status and the SSI busy status.

SSI Status (SSISR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x00C
Type RO, reset 0x0000.0003

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	BSY	RO	0	SSI Busy Bit Value Description 0 The SSI is idle. 1 The SSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.
3	RFF	RO	0	SSI Receive FIFO Full Value Description 0 The receive FIFO is not full. 1 The receive FIFO is full.
2	RNE	RO	0	SSI Receive FIFO Not Empty Value Description 0 The receive FIFO is empty. 1 The receive FIFO is not empty.
1	TNF	RO	1	SSI Transmit FIFO Not Full Value Description 0 The transmit FIFO is full. 1 The transmit FIFO is not full.

Bit/Field	Name	Type	Reset	Description
0	TFE	RO	1	SSI Transmit FIFO Empty Value Description 0 The transmit FIFO is not empty. 1 The transmit FIFO is empty.

Register 5: SSI Clock Prescale (SSICPSR), offset 0x010

The **SSICPSR** register specifies the division factor which is used to derive the `SSIClk` from the system clock. The clock is further divided by a value from 1 to 256, which is 1 + `SCR`. `SCR` is programmed in the **SSICR0** register. The frequency of the `SSIClk` is defined by:

```
SSIClk = SysClk / (CPSDVSR * (1 + SCR))
```

The value programmed into this register must be an even number between 2 and 254. The least-significant bit of the programmed number is hard-coded to zero. If an odd number is written to this register, data read back from this register has the least-significant bit as zero.

SSI Clock Prescale (SSICPSR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x010
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CPSDVSR							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CPSDVSR	R/W	0x00	SSI Clock Prescale Divisor This value must be an even number from 2 to 254, depending on the frequency of `SSIClk`. The LSB always returns 0 on reads.

Register 6: SSI Interrupt Mask (SSIIM), offset 0x014

The **SSIIM** register is the interrupt mask set or clear register. It is a read/write register and all bits are cleared on reset.

On a read, this register gives the current value of the mask on the corresponding interrupt. Setting a bit sets the mask, preventing the interrupt from being signaled to the interrupt controller. Clearing a bit clears the corresponding mask, enabling the interrupt to be sent to the interrupt controller.

SSI Interrupt Mask (SSIIM)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x014
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												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	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXIM	R/W	0	SSI Transmit FIFO Interrupt Mask Value Description 0 The transmit FIFO interrupt is masked. 1 The transmit FIFO interrupt is not masked.
2	RXIM	R/W	0	SSI Receive FIFO Interrupt Mask Value Description 0 The receive FIFO interrupt is masked. 1 The receive FIFO interrupt is not masked.
1	RTIM	R/W	0	SSI Receive Time-Out Interrupt Mask Value Description 0 The receive FIFO time-out interrupt is masked. 1 The receive FIFO time-out interrupt is not masked.
0	RORIM	R/W	0	SSI Receive Overrun Interrupt Mask Value Description 0 The receive FIFO overrun interrupt is masked. 1 The receive FIFO overrun interrupt is not masked.

Register 7: SSI Raw Interrupt Status (SSIRIS), offset 0x018

The **SSIRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

SSI Raw Interrupt Status (SSIRIS)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x018
Type RO, reset 0x0000.0008

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXRIS	RO	1	SSI Transmit FIFO Raw Interrupt Status Value Description 0 No interrupt. 1 If the `EOT` bit in the **SSICR1** register is clear, the transmit FIFO is half full or less. If the `EOT` bit is set, the transmit FIFO is empty, and the last bit has been transmitted out of the serializer. This bit is cleared when the transmit FIFO is more than half full (if the `EOT` bit is clear) or when it has any data in it (if the `EOT` bit is set).
2	RXRIS	RO	0	SSI Receive FIFO Raw Interrupt Status Value Description 0 No interrupt. 1 The receive FIFO is half full or more. This bit is cleared when the receive FIFO is less than half full.
1	RTRIS	RO	0	SSI Receive Time-Out Raw Interrupt Status Value Description 0 No interrupt. 1 The receive time-out has occurred. This bit is cleared when a 1 is written to the `RTIC` bit in the **SSI Interrupt Clear (SSIICR)** register.

Bit/Field	Name	Type	Reset	Description
0	RORRIS	RO	0	SSI Receive Overrun Raw Interrupt Status Value Description 0 No interrupt. 1 The receive FIFO has overflowed This bit is cleared when a 1 is written to the `RORIC` bit in the **SSI Interrupt Clear (SSIICR)** register.

Register 8: SSI Masked Interrupt Status (SSIMIS), offset 0x01C

The **SSIMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

SSI Masked Interrupt Status (SSIMIS)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x01C
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												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	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXMIS	RO	0	SSI Transmit FIFO Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to the transmit FIFO being half full or less (if the `EOT` bit is clear) or due to the transmission of the last data bit (if the `EOT` bit is set). This bit is cleared when the transmit FIFO is more than half full (if the `EOT` bit is clear) or when it has any data in it (if the `EOT` bit is set).
2	RXMIS	RO	0	SSI Receive FIFO Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to the receive FIFO being half full or less. This bit is cleared when the receive FIFO is less than half full.
1	RTMIS	RO	0	SSI Receive Time-Out Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to the receive time out. This bit is cleared when a 1 is written to the `RTIC` bit in the **SSI Interrupt Clear (SSIICR)** register.

Bit/Field	Name	Type	Reset	Description
0	RORMIS	RO	0	SSI Receive Overrun Masked Interrupt Status Value Description 0 An interrupt has not occurred or is masked. 1 An unmasked interrupt was signaled due to the receive FIFO overflowing. This bit is cleared when a 1 is written to the `RORIC` bit in the **SSI Interrupt Clear (SSIICR)** register.

Register 9: SSI Interrupt Clear (SSIICR), offset 0x020

The **SSIICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

SSI Interrupt Clear (SSIICR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x020
Type W1C, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														RTIC	RORIC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	RTIC	W1C	0	SSI Receive Time-Out Interrupt Clear Writing a 1 to this bit clears the `RTRIS` bit in the **SSIRIS** register and the `RTMIS` bit in the **SSIMIS** register.
0	RORIC	W1C	0	SSI Receive Overrun Interrupt Clear Writing a 1 to this bit clears the `RORRIS` bit in the **SSIRIS** register and the `RORMIS` bit in the **SSIMIS** register.

Register 10: SSI DMA Control (SSIDMACTL), offset 0x024

The **SSIDMACTL** register is the μDMA control register.

SSI DMA Control (SSIDMACTL)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x024
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														TXDMAE	RXDMAE
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	TXDMAE	R/W	0	Transmit DMA Enable Value Description 0 μDMA for the transmit FIFO is disabled. 1 μDMA for the transmit FIFO is enabled.
0	RXDMAE	R/W	0	Receive DMA Enable Value Description 0 μDMA for the receive FIFO is disabled. 1 μDMA for the receive FIFO is enabled.

Register 11: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0

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

SSI Peripheral Identification 4 (SSIPeriphID4)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFD0
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	SSI Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

Register 12: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4

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

SSI Peripheral Identification 5 (SSIPeriphID5)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFD4
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	SSI Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 13: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8

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

SSI Peripheral Identification 6 (SSIPeriphID6)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFD8
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID6							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	SSI Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 14: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC

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

SSI Peripheral Identification 7 (SSIPeriphID7)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFDC
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	SSI Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 15: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0

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

SSI Peripheral Identification 0 (SSIPeriphID0)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFE0
Type RO, reset 0x0000.0022

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x22	SSI Peripheral ID Register [7:0] Can be used by software to identify the presence of this peripheral.

Register 16: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4

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

SSI Peripheral Identification 1 (SSIPeriphID1)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFE4
Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	SSI Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 17: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8

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

SSI Peripheral Identification 2 (SSIPeriphID2)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFE8
Type RO, reset 0x0000.0018

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	SSI Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 18: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC

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

SSI Peripheral Identification 3 (SSIPeriphID3)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFEC
Type RO, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	SSI Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 19: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0

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

SSI PrimeCell Identification 0 (SSIPCellID0)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFF0
Type RO, reset 0x0000.000D

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	SSI PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

Register 20: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4

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

SSI PrimeCell Identification 1 (SSIPCellID1)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFF4
Type RO, reset 0x0000.00F0

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	SSI PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

Register 21: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8

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

SSI PrimeCell Identification 2 (SSIPCellID2)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFF8
Type RO, reset 0x0000.0005

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	SSI PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

Register 22: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC

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

SSI PrimeCell Identification 3 (SSIPCellID3)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFFC
Type RO, reset 0x0000.00B1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	SSI PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

16 Inter-Integrated Circuit (I^2C) Interface

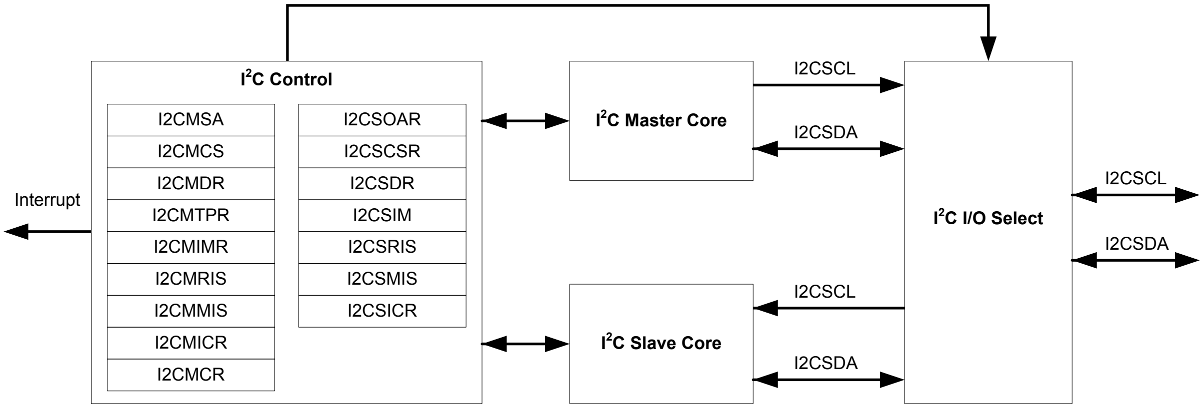
The Inter-Integrated Circuit (I^2C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL), and interfaces to external I^2C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I^2C bus may also be used for system testing and diagnostic purposes in product development and manufacture. The LM3S1W16 microcontroller includes two I^2C modules, providing the ability to interact (both transmit and receive) with other I^2C devices on the bus.

The Stellaris® LM3S1W16 controller includes two I^2C modules with the following features:

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

16.1 Block Diagram

Figure 16-1. I²C Block Diagram



16.2 Signal Description

Table 16-1 on page 578 lists the external signals of the I²C interface and describes the function of each. The I²C interface signals are alternate functions for some GPIO signals and default to be GPIO signals at reset., with the exception of the `I2C0SCL` and `I2CSDA` pins which default to the I²C function. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the I²C signals. The `AFSEL` bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 307) should be set to choose the I²C function. The number in parentheses is the encoding that must be programmed into the `PMCn` field in the **GPIO Port Control (GPIOPCTL)** register (page 324) to assign the I²C signal to the specified GPIO port pin. Note that the I²C pins should be set to open drain using the **GPIO Open Drain Select (GPIOODR)** register. For more information on configuring GPIOs, see "General-Purpose Input/Outputs (GPIOs)" on page 285.

Table 16-1. Signals for I2C

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`I2C0SCL`	47	PB2 (1)	I/O	OD	I²C module 0 clock.
`I2C0SDA`	27	PB3 (1)	I/O	OD	I²C module 0 data.
`I2C1SCL`	17 25	PA0 (8) PA6 (1)	I/O	OD	I²C module 1 clock.
`I2C1SDA`	18 26	PA1 (8) PA7 (1)	I/O	OD	I²C module 1 data.

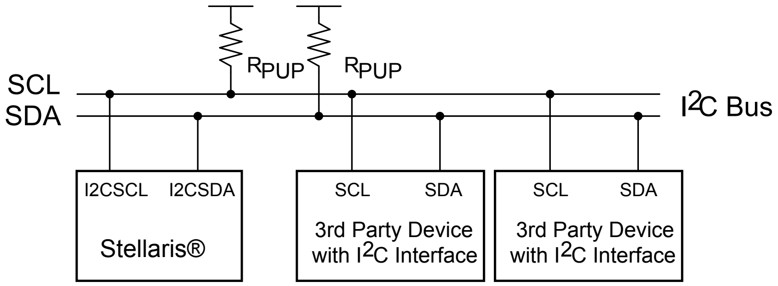
a. The TTL designation indicates the pin has TTL-compatible voltage levels.

16.3 Functional Description

Each I²C module is comprised of both master and slave functions which are implemented as separate peripherals. For proper operation, the SDA and SCL pins must be connected to bi-directional open-drain pads. A typical I²C bus configuration is shown in Figure 16-2.

See "Inter-Integrated Circuit (I²C) Interface" on page 656 for I²C timing diagrams.

Figure 16-2. I^{2}C Bus Configuration



16.3.1 I^{2}C Bus Functional Overview

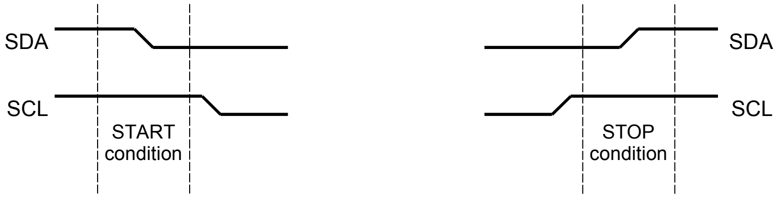
The I^{2}C bus uses only two signals: SDA and SCL, named `I2CSDA` and `I2CSCL` on Stellaris® microcontrollers. SDA is the bi-directional serial data line and SCL is the bi-directional serial clock line. The bus is considered idle when both lines are High.

Every transaction on the I^{2}C bus is nine bits long, consisting of eight data bits and a single acknowledge bit. The number of bytes per transfer (defined as the time between a valid START and STOP condition, described in “START and STOP Conditions” on page 579) is unrestricted, but each byte has to be followed by an acknowledge bit, and data must be transferred MSB first. When a receiver cannot receive another complete byte, it can hold the clock line SCL Low and force the transmitter into a wait state. The data transfer continues when the receiver releases the clock SCL.

16.3.1.1 START and STOP Conditions

The protocol of the I^{2}C bus defines two states to begin and end a transaction: START and STOP. A High-to-Low transition on the SDA line while the SCL is High is defined as a START condition, and a Low-to-High transition on the SDA line while SCL is High is defined as a STOP condition. The bus is considered busy after a START condition and free after a STOP condition. See Figure 16-3.

Figure 16-3. START and STOP Conditions



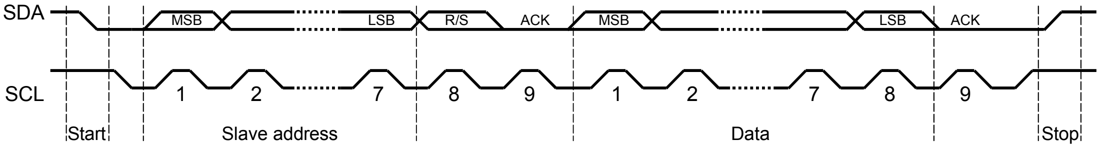
The `STOP` bit determines if the cycle stops at the end of the data cycle or continues on to a repeated START condition. To generate a single transmit cycle, the **I^{2}C Master Slave Address (I2CMSA)** register is written with the desired address, the `R/S` bit is cleared, and the Control register is written with `ACK`=X (0 or 1), `STOP=1`, `START=1`, and `RUN=1` to perform the operation and stop. When the operation is completed (or aborted due an error), the interrupt pin becomes active and the data may be read from the **I2CMDR** register. When the I^{2}C module operates in Master receiver mode, the `ACK` bit is nornally set causing the I^{2}C bus controller to transmit an acknowledge automatically after each byte. This bit must be cleared when the I^{2}C bus controller requires no further data to be transmitted from the slave transmitter.

When operating in slave mode, two bits in the **I2CSRIS** register indicate detection of start and stop conditions on the bus; while two bits in the **I2CSMIS** register allow start and stop conditions to be promoted to controller interrupts (when interrupts are enabled).

16.3.1.2 Data Format with 7-Bit Address

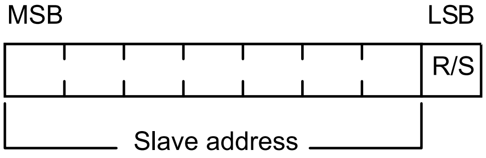
Data transfers follow the format shown in Figure 16-4. After the START condition, a slave address is transmitted. This address is 7-bits long followed by an eighth bit, which is a data direction bit (`R/S` bit in the **I2CMSA** register). If the `R/S` bit is clear, it indicates a transmit operation (send), and if it is set, it indicates a request for data (receive). A data transfer is always terminated by a STOP condition generated by the master, however, a master can initiate communications with another device on the bus by generating a repeated START condition and addressing another slave without first generating a STOP condition. Various combinations of receive/transmit formats are then possible within a single transfer.

Figure 16-4. Complete Data Transfer with a 7-Bit Address



The first seven bits of the first byte make up the slave address (see Figure 16-5). The eighth bit determines the direction of the message. A zero in the R/S position of the first byte means that the master transmits (sends) data to the selected slave, and a one in this position means that the master receives data from the slave.

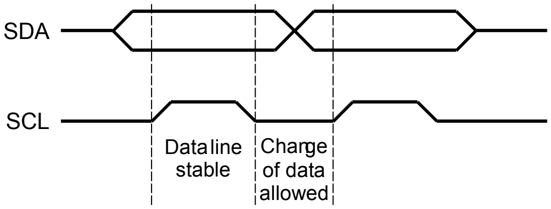
Figure 16-5. R/S Bit in First Byte



16.3.1.3 Data Validity

The data on the SDA line must be stable during the high period of the clock, and the data line can only change when SCL is Low (see Figure 16-6).

Figure 16-6. Data Validity During Bit Transfer on the I^2C Bus



16.3.1.4 Acknowledge

All bus transactions have a required acknowledge clock cycle that is generated by the master. During the acknowledge cycle, the transmitter (which can be the master or slave) releases the SDA line. To acknowledge the transaction, the receiver must pull down SDA during the acknowledge clock cycle. The data transmitted out by the receiver during the acknowledge cycle must comply with the data validity requirements described in “Data Validity” on page 580.

When a slave receiver does not acknowledge the slave address, SDA must be left High by the slave so that the master can generate a STOP condition and abort the current transfer. If the master device is acting as a receiver during a transfer, it is responsible for acknowledging each transfer made by the slave. Because the master controls the number of bytes in the transfer, it signals the end of data to the slave transmitter by not generating an acknowledge on the last data byte. The slave transmitter must then release SDA to allow the master to generate the STOP or a repeated START condition.

16.3.1.5 Arbitration

A master may start a transfer only if the bus is idle. It's possible for two or more masters to generate a START condition within minimum hold time of the START condition. In these situations, an arbitration scheme takes place on the SDA line, while SCL is High. During arbitration, the first of the competing master devices to place a '1' (High) on SDA while another master transmits a '0' (Low) switches off its data output stage and retires until the bus is idle again.

Arbitration can take place over several bits. Its first stage is a comparison of address bits, and if both masters are trying to address the same device, arbitration continues on to the comparison of data bits.

16.3.2 Available Speed Modes

The I^2C bus can run in either Standard mode (100 kbps) or Fast mode (400 kbps). The selected mode should match the speed of the other I^2C devices on the bus. The mode is selected by using a value in the **I^2C Master Timer Period (I2CMTPR)** register that results in an SCL frequency of 100 kbps for Standard mode or 400 kbps for Fast mode.

The I^2C clock rate is determined by the parameters *CLK_PRD*, *TIMER_PRD*, *SCL_LP*, and *SCL_HP* where:

CLK_PRD is the system clock period

SCL_LP is the low phase of SCL (fixed at 6)

SCL_HP is the high phase of SCL (fixed at 4)

TIMER_PRD is the programmed value in the **I2CMTPR** register (see page 600).

The I^2C clock period is calculated as follows:

$SCL\_PERIOD = 2 \times (1 + TIMER\_PRD) \times (SCL\_LP + SCL\_HP) \times CLK\_PRD$

For example:

$CLK\_PRD = 50$ ns

$TIMER\_PRD = 2$

$SCL\_LP=6$

$SCL\_HP=4$

yields a SCL frequency of:

$1/SCL\_PERIOD = 333$ Khz

Table 16-2 gives examples of the timer periods that should be used to generate both Standard and Fast mode SCL frequencies based on various system clock frequencies.

Table 16-2. Examples of I^2C Master Timer Period versus Speed Mode

System Clock	Timer Period	Standard Mode	Timer Period	Fast Mode
4 MHz	0x01	100 Kbps	-	-
6 MHz	0x02	100 Kbps	-	-
12.5 MHz	0x06	89 Kbps	0x01	312 Kbps
16.7 MHz	0x08	93 Kbps	0x02	278 Kbps
20 MHz	0x09	100 Kbps	0x02	333 Kbps
25 MHz	0x0C	96.2 Kbps	0x03	312 Kbps
33 MHz	0x10	97.1 Kbps	0x04	330 Kbps
40 MHz	0x13	100 Kbps	0x04	400 Kbps
50 MHz	0x18	100 Kbps	0x06	357 Kbps

16.3.3 Interrupts

The I^2C can generate interrupts when the following conditions are observed:

- Master transaction completed
- Master transaction error
- Slave transaction received
- Slave transaction requested
- Stop condition on bus detected
- Start condition on bus detected

The I^2C master and I^2C slave modules have separate interrupt signals. While both modules can generate interrupts for multiple conditions, only a single interrupt signal is sent to the interrupt controller.

16.3.3.1 I^2C Master Interrupts

The I^2C master module generates an interrupt when a transaction completes (either transmit or receive), or when an error occurs during a transaction. To enable the I^2C master interrupt, software must set the `IM` bit in the **I^2C Master Interrupt Mask (I2CMIMR)** register. When an interrupt condition is met, software must check the `ERROR` bit in the **I^2C Master Control/Status (I2CMCS)** register to verify that an error didn't occur during the last transaction. An error condition is asserted if the last transaction wasn't acknowledged by the slave, or if the master was forced to give up ownership of the bus due to a lost arbitration round with another master. If an error is not detected, the application can proceed with the transfer. The interrupt is cleared by writing a 1 to the `IC` bit in the **I^2C Master Interrupt Clear (I2CMICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I^2C Master Raw Interrupt Status (I2CMRIS)** register.

16.3.3.2 I^2C Slave Interrupts

The slave module can generate an interrupt when data has been received or requested. This interrupt is enabled by setting the `DATAIM` bit in the **I^2C Slave Interrupt Mask (I2CSIMR)** register. Software determines whether the module should write (transmit) or read (receive) data from the **I^2C Slave Data (I2CSDR)** register, by checking the `RREQ` and `TREQ` bits of the **I^2C Slave Control/Status**

(I2CSCSR) register. If the slave module is in receive mode and the first byte of a transfer is received, the `FBR` bit is set along with the `RREQ` bit. The interrupt is cleared by setting the `DATAIC` bit in the **I²C Slave Interrupt Clear (I2CSICR)** register.

In addition, the slave module can generate an interrupt when a start and stop condition is detected. These interrupts are enabled by setting the `STARTIM` and `STOPIM` bits of the **I²C Slave Interrupt Mask (I2CSIMR)** register and cleared by writing a 1 to the `STOPIC` and `STARTIC` bits of the **I²C Slave Interrupt Clear (I2CSICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I²C Slave Raw Interrupt Status (I2CSRIS)** register.

16.3.4 Loopback Operation

The I²C modules can be placed into an internal loopback mode for diagnostic or debug work by setting the `LPBK` bit in the **I²C Master Configuration (I2CMCR)** register. In loopback mode, the SDA and SCL signals from the master and slave modules are tied together.

16.3.5 Command Sequence Flow Charts

This section details the steps required to perform the various I²C transfer types in both master and slave mode.

16.3.5.1 I²C Master Command Sequences

The figures that follow show the command sequences available for the I²C master.

Figure 16-7. Master Single TRANSMIT

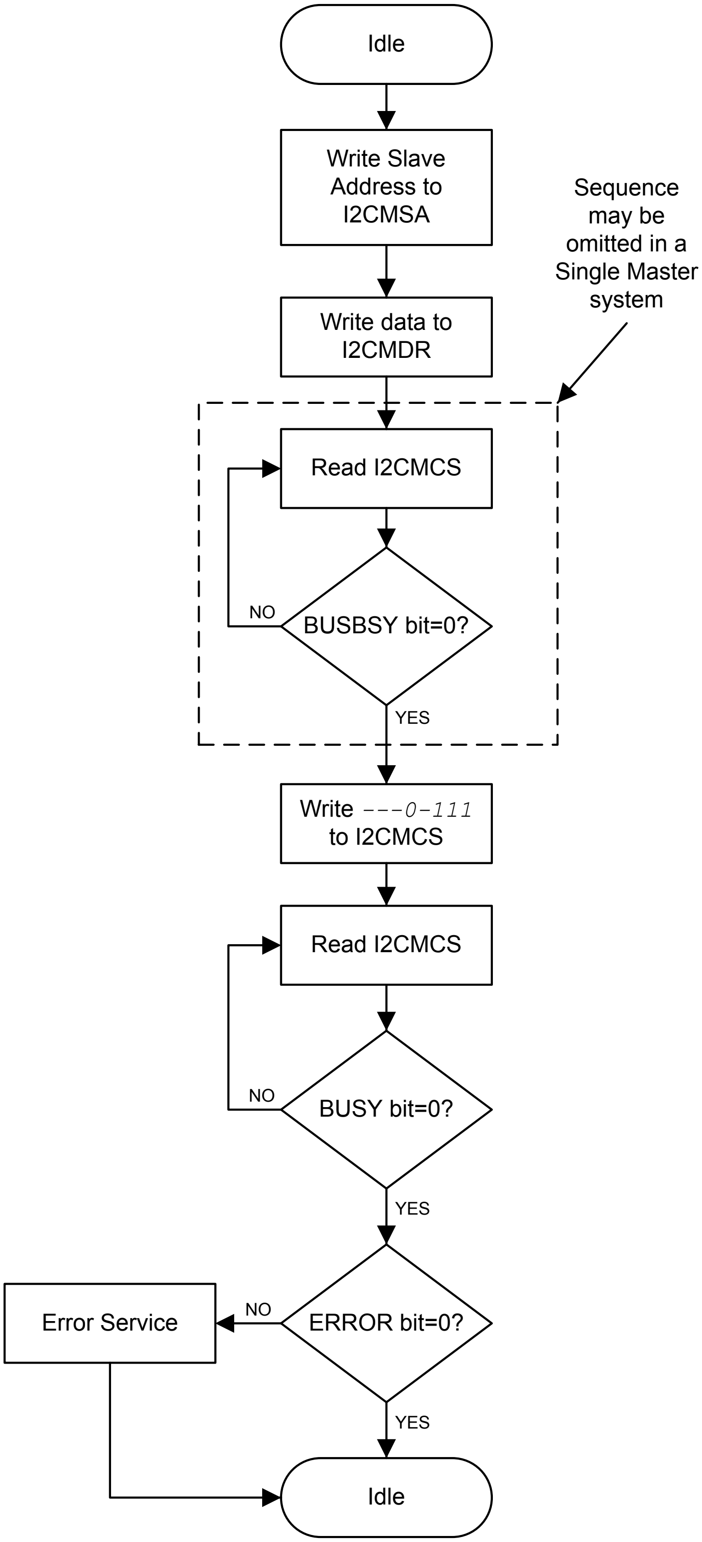


Figure 16-8. Master Single RECEIVE

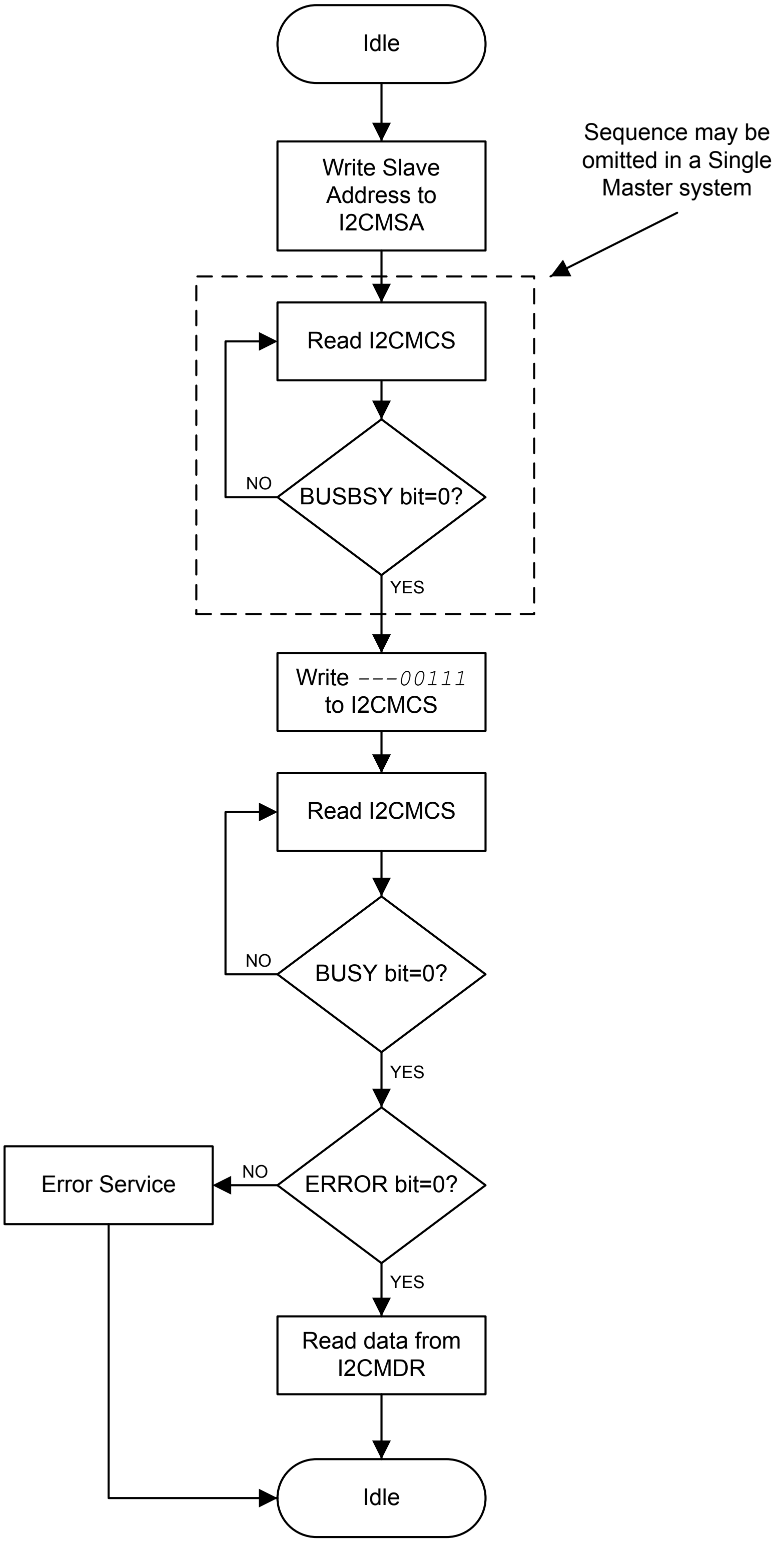


Figure 16-9. Master TRANSMIT with Repeated START

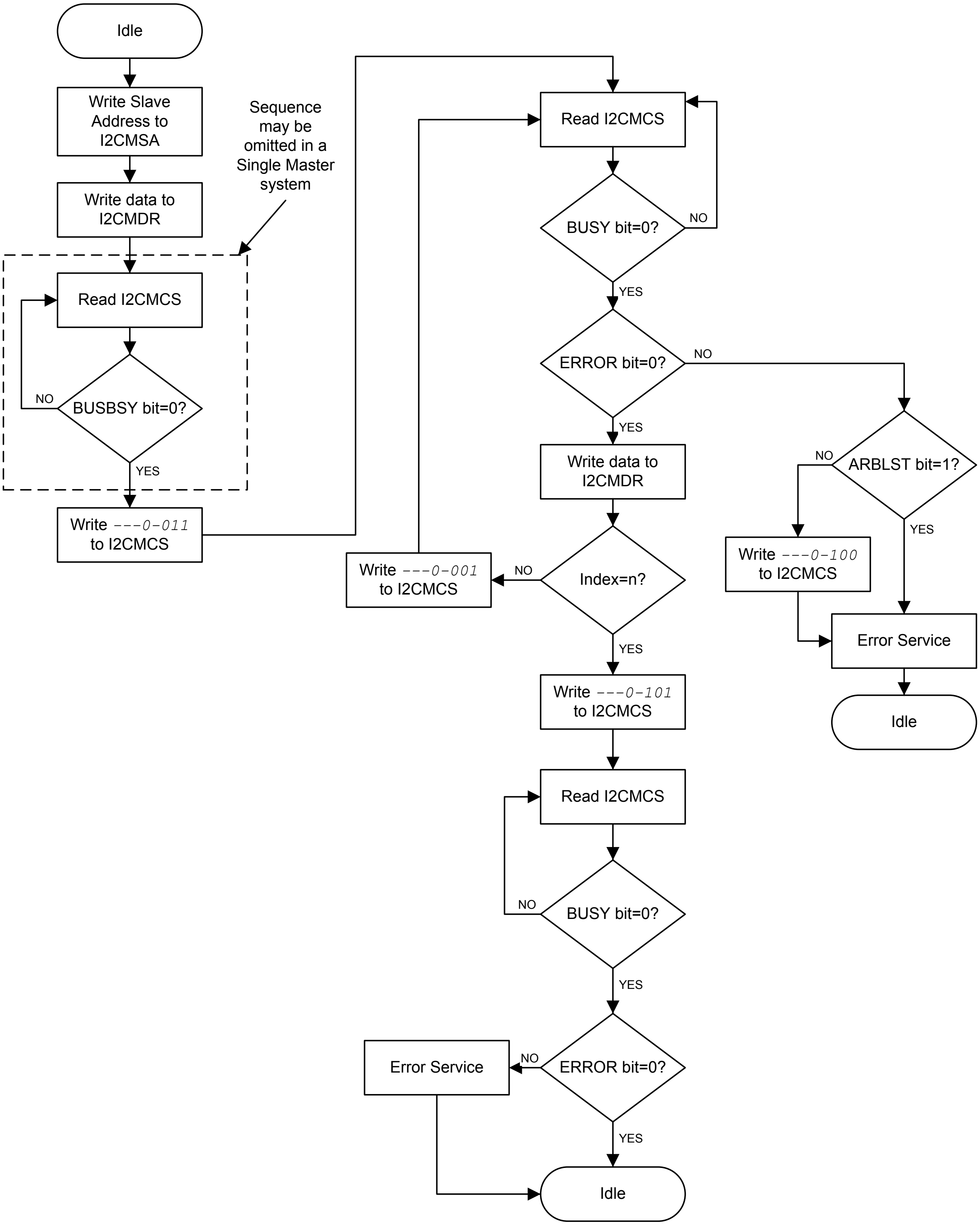


Figure 16-10. Master RECEIVE with Repeated START

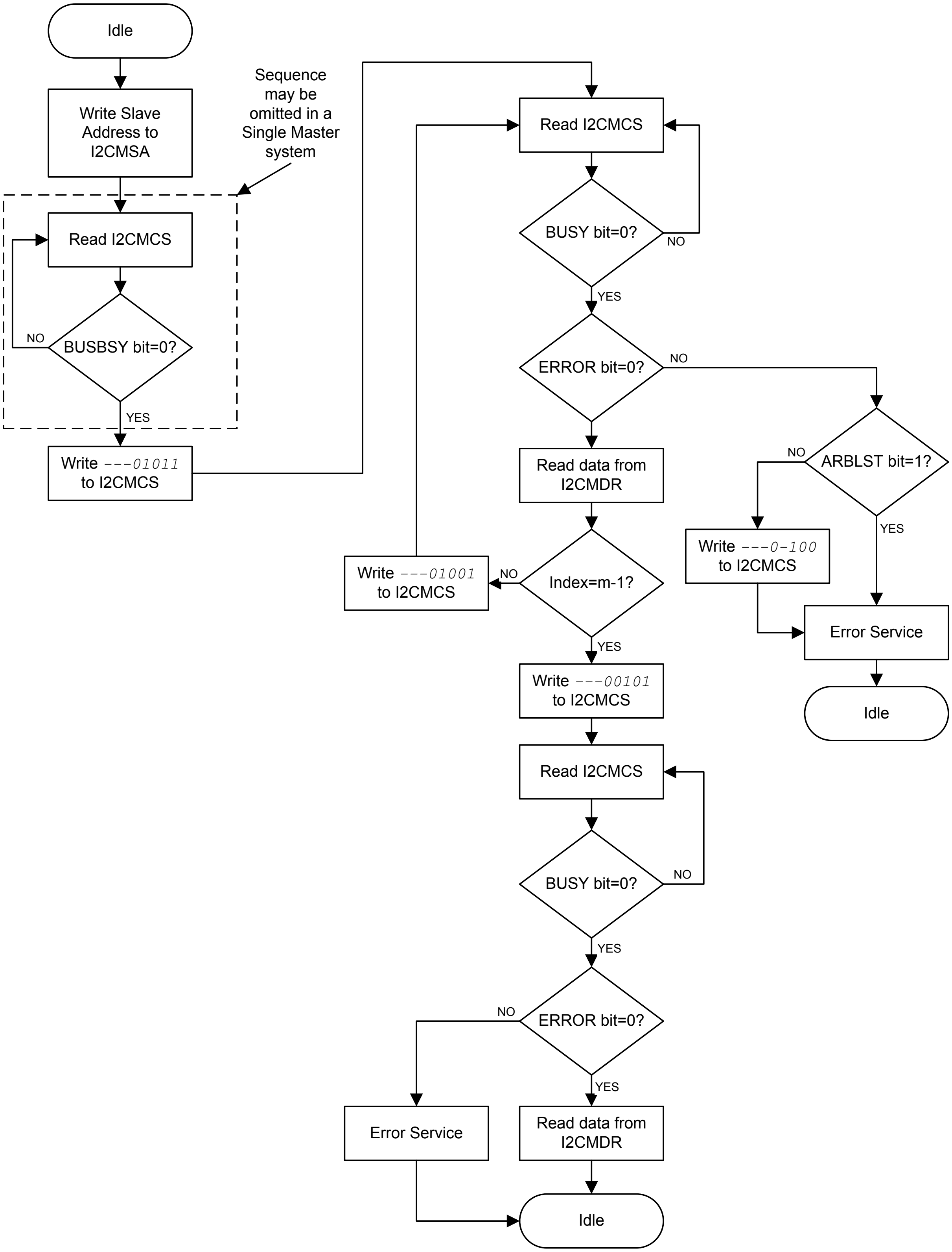


Figure 16-11. Master RECEIVE with Repeated START after TRANSMIT with Repeated START

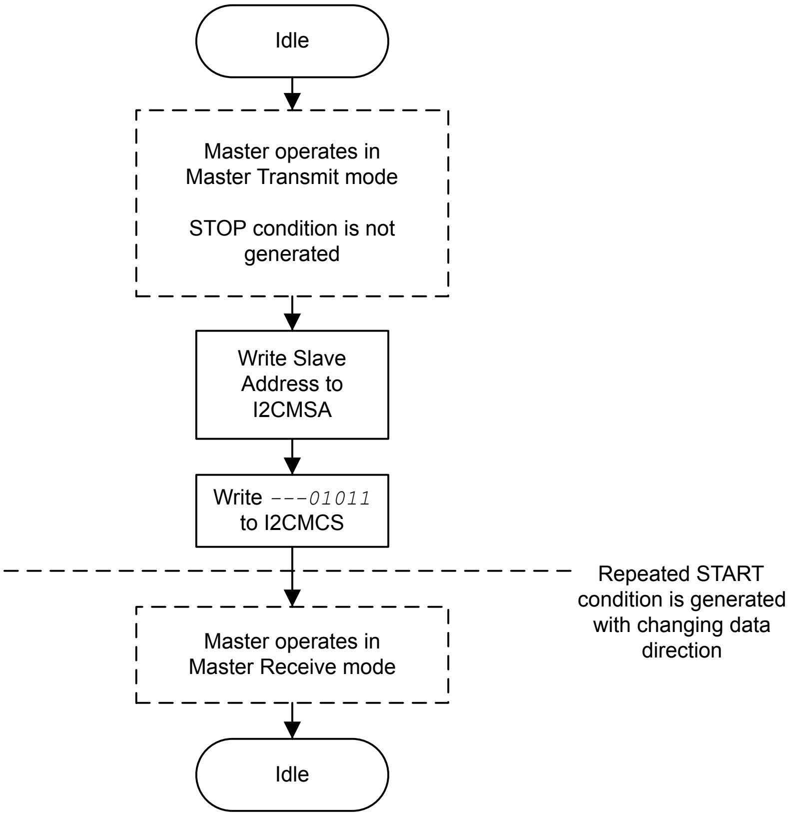
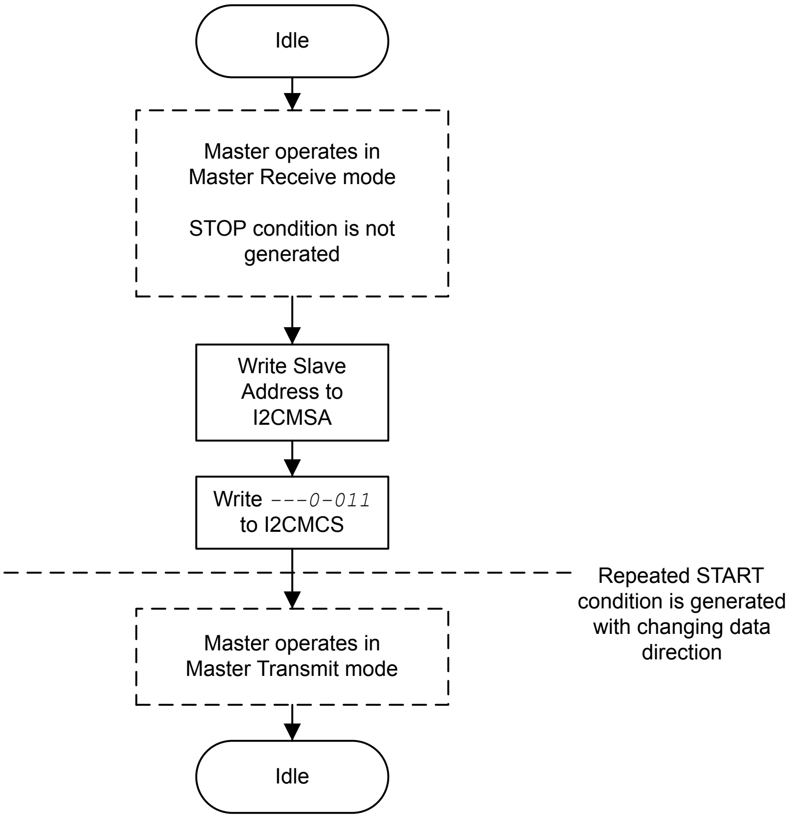


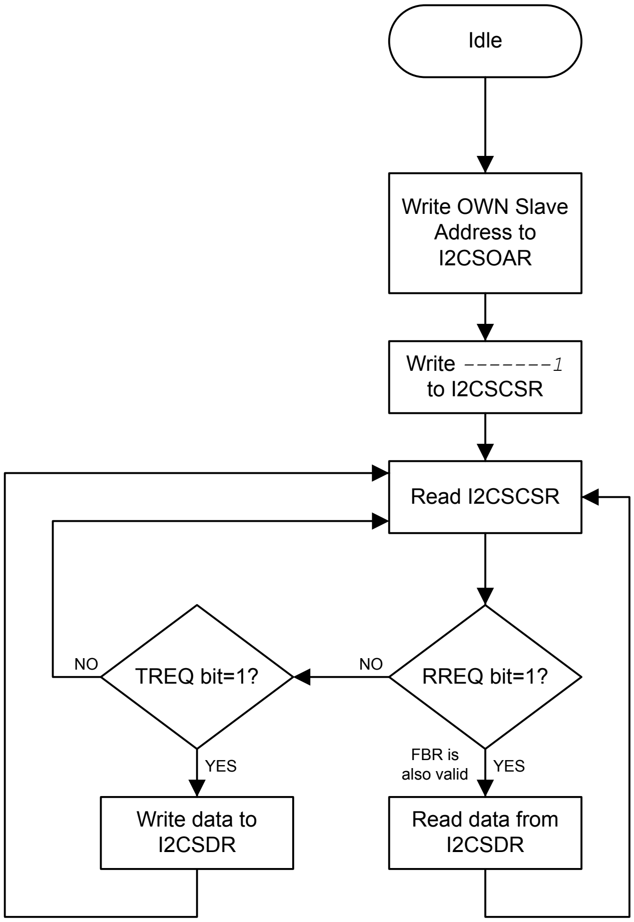
Figure 16-12. Master TRANSMIT with Repeated START after RECEIVE with Repeated START



16.3.5.2 I²C Slave Command Sequences

Figure 16-13 on page 590 presents the command sequence available for the I²C slave.

Figure 16-13. Slave Command Sequence



16.4 Initialization and Configuration

The following example shows how to configure the I^2C module to transmit a single byte as a master. This assumes the system clock is 20 MHz.

1. Enable the I^2C clock by writing a value of 0x0000.1000 to the **RCGC1** register in the System Control module (see page 147).
2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module (see page 156). To find out which GPIO port to enable, refer to Table 19-5 on page 640.
3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register (see page 307). To determine which GPIOs to configure, see Table 19-4 on page 636.
4. Enable the I^2C pins for Open Drain operation. See page 312.
5. Configure the `PMCn` fields in the **GPIOPCTL** register to assign the I^2C signals to the appropriate pins. See page 324 and Table 19-5 on page 640.
6. Initialize the I^2C Master by writing the **I2CMCR** register with a value of 0x0000.0010.

7. 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.

8. Specify the slave address of the master and that the next operation is a Transmit by writing the **I2CMSA** register with a value of 0x0000.0076. This sets the slave address to 0x3B.

9. Place data (byte) to be transmitted in the data register by writing the **I2CMDR** register with the desired data.

10. Initiate a single byte transmit of the data from Master to Slave by writing the **I2CMCS** register with a value of 0x0000.0007 (STOP, START, RUN).

11. Wait until the transmission completes by polling the **I2CMCS** register's BUSBSY bit until it has been cleared.

16.5 Register Map

Table 16-3 on page 591 lists the I^{2}C registers. All addresses given are relative to the I^{2}C base addresses for the master and slave:

- I^{2}C Master 0: 0x4002.0000
- I^{2}C Slave 0: 0x4002.0800
- I^{2}C Master 1: 0x4002.1000
- I^{2}C Slave 1: 0x4002.1800

Note that the I^{2}C module clock must be enabled before the registers can be programmed (see page 147).

Table 16-3. Inter-Integrated Circuit (I^{2}C) Interface Register Map

Offset	Name	Type	Reset	Description	See page
I^{2}C Master					
0x000	I2CMSA	R/W	0x0000.0000	I2C Master Slave Address	593
0x004	I2CMCS	R/W	0x0000.0000	I2C Master Control/Status	594
0x008	I2CMDR	R/W	0x0000.0000	I2C Master Data	599
0x00C	I2CMTPR	R/W	0x0000.0001	I2C Master Timer Period	600
0x010	I2CMIMR	R/W	0x0000.0000	I2C Master Interrupt Mask	601
0x014	I2CMRIS	RO	0x0000.0000	I2C Master Raw Interrupt Status	602
0x018	I2CMMIS	RO	0x0000.0000	I2C Master Masked Interrupt Status	603
0x01C	I2CMICR	WO	0x0000.0000	I2C Master Interrupt Clear	604
0x020	I2CMCR	R/W	0x0000.0000	I2C Master Configuration	605

Table 16-3. Inter-Integrated Circuit (I^2C) Interface Register Map *(continued)*

Offset	Name	Type	Reset	Description	See page
I^2C Slave					
0x000	I2CSOAR	R/W	0x0000.0000	I2C Slave Own Address	606
0x004	I2CSCSR	RO	0x0000.0000	I2C Slave Control/Status	607
0x008	I2CSDR	R/W	0x0000.0000	I2C Slave Data	609
0x00C	I2CSIMR	R/W	0x0000.0000	I2C Slave Interrupt Mask	610
0x010	I2CSRIS	RO	0x0000.0000	I2C Slave Raw Interrupt Status	611
0x014	I2CSMIS	RO	0x0000.0000	I2C Slave Masked Interrupt Status	612
0x018	I2CSICR	WO	0x0000.0000	I2C Slave Interrupt Clear	613

16.6 Register Descriptions (I^2C Master)

The remainder of this section lists and describes the I^2C master registers, in numerical order by address offset. See also “Register Descriptions (I^2C Slave)” on page 605.

Register 1: I^2C Master Slave Address (I2CMSA), offset 0x000

This register consists of eight bits: seven address bits (A6-A0), and a Receive/Send bit, which determines if the next operation is a Receive (High), or Transmit (Low).

I2C Master Slave Address (I2CMSA)

I2C 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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:1	SA	R/W	0x00	I^2C Slave Address This field specifies bits A6 through A0 of the slave address.
0	R/S	R/W	0	Receive/Send The `R/S` bit specifies if the next operation is a Receive (High) or Transmit (Low).

Value	Description
0	Transmit
1	Receive

Register 2: I^{2}C Master Control/Status (I2CMCS), offset 0x004

This register accesses seven status bits when read and four control bits when written.

The status register consists of seven bits, which when read determine the state of the I^{2}C bus controller.

The control register consists of four bits: the `RUN`, `START`, `STOP`, and `ACK` bits. The `START` bit generates the START or REPEATED START condition.

The `STOP` bit determines if the cycle stops at the end of the data cycle or continues on to a repeated START condition. To generate a single transmit cycle, the **I^{2}C Master Slave Address (I2CMSA)** register is written with the desired address, the `R/S` bit is cleared, and the Control register is written with `ACK`=X (0 or 1), `STOP`=1, `START`=1, and `RUN`=1 to perform the operation and stop. When the operation is completed (or aborted due an error), the interrupt pin becomes active and the data may be read from the **I2CMDR** register. When the I^{2}C module operates in Master receiver mode, the `ACK` bit is nornally set causing the I^{2}C bus controller to transmit an acknowledge automatically after each byte. This bit must be cleared when the I^{2}C bus controller requires no further data to be transmitted 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 RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									BUSBSY	IDLE	ARBLST	DATACK	ADRACK	ERROR	BUSY
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	BUSBSY	RO	0	Bus Busy Value Description 0 The I^{2}C bus is idle. 1 The I^{2}C bus is busy. The bit changes based on the START and STOP conditions.
5	IDLE	RO	0	I^{2}C Idle Value Description 0 The I^{2}C controller is not idle. 1 The I^{2}C controller is idle.

Bit/Field	Name	Type	Reset	Description
4	ARBLST	RO	0	Arbitration Lost Value Description 0 The I^{2}C controller won arbitration. 1 The I^{2}C controller lost arbitration.
3	DATACK	RO	0	Acknowledge Data Value Description 0 The transmitted data was acknowledged 1 The transmitted data was not acknowledged.
2	ADRACK	RO	0	Acknowledge Address Value Description 0 The transmitted address was acknowledged 1 The transmitted address was not acknowledged.
1	ERROR	RO	0	Error Value Description 0 No error was detected on the last operation. 1 An error occurred on the last operation. The error can be from the slave address not being acknowledged, the transmit data not being acknowledged, or because the controller lost arbitration.
0	BUSY	RO	0	I^{2}C Busy Value Description 0 The controller is idle. 1 The controller is busy. When the BUSY bit is set, the other status bits are not valid.

Write-Only Control Register

I2C Master Control/Status (I2CMCS)

I2C Master 0 base: 0x4002.0000
I2C Master 1 base: 0x4002.1000
Offset 0x004
Type WO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												ACK	STOP	START	RUN
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	WO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ACK	WO	0	Data Acknowledge Enable Value Description 0 The received data byte is not acknowledged automatically by the master. 1 The received data byte is acknowledged automatically by the master. See field decoding in Table 16-4 on page 597.
2	STOP	WO	0	Generate STOP Value Description 0 The controller does not generate the STOP condition. 1 The controller generates the STOP condition. See field decoding in Table 16-4 on page 597.
1	START	WO	0	Generate START Value Description 0 The controller does not generate the START condition. 1 The controller generates the START or repeated START condition. See field decoding in Table 16-4 on page 597.
0	RUN	WO	0	I^2C Master Enable Value Description 0 The master is disabled. 1 The master is enabled to transmit or receive data. See field decoding in Table 16-4 on page 597.

Table 16-4. Write Field Decoding for I2CMCS[3:0] Field

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 TRANSMIT (master goes to the Master Transmit state).
	0	X	1	1	1	START condition followed by a TRANSMIT and STOP condition (master remains in Idle state).
	1	0	0	1	1	START condition followed by RECEIVE operation with negative ACK (master goes to the Master Receive state).
	1	0	1	1	1	START condition followed by RECEIVE and STOP condition (master remains in Idle state).
	1	1	0	1	1	START condition followed by RECEIVE (master goes to the Master Receive state).
	1	1	1	1	1	Illegal
	All other combinations not listed are non-operations.					NOP
Master Transmit	X	X	0	0	1	TRANSMIT operation (master remains in Master Transmit state).
	X	X	1	0	0	STOP condition (master goes to Idle state).
	X	X	1	0	1	TRANSMIT followed by STOP condition (master goes to Idle state).
	0	X	0	1	1	Repeated START condition followed by a TRANSMIT (master remains in Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by TRANSMIT and STOP condition (master goes to Idle state).
	1	0	0	1	1	Repeated START condition followed by a RECEIVE operation with a negative ACK (master goes to Master Receive state).
	1	0	1	1	1	Repeated START condition followed by a TRANSMIT and STOP condition (master goes to Idle state).
	1	1	0	1	1	Repeated START condition followed by RECEIVE (master goes to Master Receive state).
	1	1	1	1	1	Illegal.
	All other combinations not listed are non-operations.					NOP.

Table 16-4. Write Field Decoding for I2CMCS[3:0] Field *(continued)*

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 TRANSMIT (master goes to Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by TRANSMIT and STOP condition (master goes to Idle state).
	All other combinations not listed are non-operations.					NOP.

a. An X in a table cell indicates the bit can be 0 or 1.

b. In Master Receive mode, a STOP condition should be generated only after a Data Negative Acknowledge executed by the master or an Address Negative Acknowledge executed by the slave.

Register 3: I^{2}C Master Data (I2CMDR), offset 0x008

Important: Use caution when reading this register. Performing a read may change bit status.

This register contains the data to be transmitted when in the Master Transmit state and the data received when in the Master Receive state.

I2C Master Data (I2CMDR)

I2C 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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x00	Data Transferred Data transferred during transaction.

Register 4: I^2C 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	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TPR	R/W	0x1	SCL Clock Period This field specifies the period of the SCL clock. $SCL\_PRD = 2 \times (1 + \mathrm{TPR}) \times (SCL\_LP + SCL\_HP) \times CLK\_PRD$ where: SCL_PRD is the SCL line period (I^2C 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). CLK_PRD is the system clock period in ns.

Register 5: I^{2}C Master Interrupt Mask (I2CMIMR), offset 0x010

This register controls whether a raw interrupt is promoted to a controller interrupt.

I2C Master Interrupt Mask (I2CMIMR)

I2C Master 0 base: 0x4002.0000
I2C Master 1 base: 0x4002.1000
Offset 0x010
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															IM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Value	Description
1	The master interrupt is sent to the interrupt controller when the `RIS` bit in the **I2CMRIS** register is set.
0	The `RIS` interrupt is suppressed and not sent to the interrupt controller.

Register 6: I^2C 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	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	RIS	RO	0	Raw Interrupt Status Value Description 1 A master interrupt is pending. 0 No interrupt. This bit is cleared by writing a 1 to the `IC` bit in the **I2CMICR** register.

Register 7: I^{2}C Master Masked Interrupt Status (I2CMMIS), offset 0x018

This register specifies whether an interrupt was signaled.

I2C Master Masked Interrupt Status (I2CMMIS)

I2C Master 0 base: 0x4002.0000
I2C Master 1 base: 0x4002.1000
Offset 0x018
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															MIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	MIS	RO	0	Masked Interrupt Status Value Description 1 An unmasked master interrupt was signaled is pending. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `IC` bit in the **I2CMICR** register.

Register 8: I^{2}C Master Interrupt Clear (I2CMICR), offset 0x01C

This register clears the raw interrupt.

I2C Master Interrupt Clear (I2CMICR)

I2C Master 0 base: 0x4002.0000
I2C Master 1 base: 0x4002.1000
Offset 0x01C
Type WO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															IC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IC	WO	0	Interrupt Clear Writing a 1 to this bit clears the `RIS` bit in the **I2CMRIS** register and the `MIS` bit in the **I2CMMIS** register. A read of this register returns no meaningful data.

Register 9: I²C Master Configuration (I2CMCR), offset 0x020

This register configures the mode (Master or Slave) and sets the interface for test mode loopback.

I2C Master Configuration (I2CMCR)

I2C 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	R/W	R/W	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SFE	R/W	0	I²C Slave Function Enable Value Description 1 Slave mode is enabled. 0 Slave mode is disabled.
4	MFE	R/W	0	I²C Master Function Enable Value Description 1 Master mode is enabled. 0 Master mode is disabled.
3:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	LPBK	R/W	0	I²C Loopback Value Description 1 The controller in a test mode loopback configuration. 0 Normal operation.

16.7 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 592.

Register 10: I^2C Slave Own Address (I2CSOAR), offset 0x000

This register consists of seven address bits that identify the Stellaris® I^2C device on the I^2C bus.

I2C Slave Own Address (I2CSOAR)

I2C Slave 0 base: 0x4002.0800
I2C Slave 1 base: 0x4002.1800
Offset 0x000
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									OAR						
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	OAR	R/W	0x00	I^2C Slave Own Address This field specifies bits A6 through A0 of the slave address.

Register 11: I²C Slave Control/Status (I2CSCSR), 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 (I2CSCSR)

I2C Slave 0 base: 0x4002.0800
I2C Slave 1 base: 0x4002.1800
Offset 0x004
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													FBR	TREQ	RREQ
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	FBR	RO	0	First Byte Received Value Description 1 The first byte following the slave's own address has been received. 0 The first byte has not been received. This bit is only valid when the `RREQ` bit is set and is automatically cleared when data has been read from the **I2CSDR** register. **Note:** This bit is not used for slave transmit operations.
1	TREQ	RO	0	Transmit Request Value Description 1 The I²C controller has been addressed as a slave transmitter and is using clock stretching to delay the master until data has been written to the **I2CSDR** register. 0 No outstanding transmit request.

Bit/Field	Name	Type	Reset	Description
0	RREQ	RO	0	Receive Request

Value	Description
1	The I^{2}C controller has outstanding receive data from the I^{2}C master and is using clock stretching to delay the master until the data has been read from the **I2CSDR** register.
0	No outstanding receive data.

Write-Only Control Register

I2C Slave Control/Status (I2CSCSR)

I2C Slave 0 base: 0x4002.0800
I2C Slave 1 base: 0x4002.1800
Offset 0x004
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															DA
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Value	Description
0	Disables the I^{2}C slave operation.
1	Enables the I^{2}C slave operation.

Register 12: I^{2}C Slave Data (I2CSDR), offset 0x008

Important: Use caution when reading this register. Performing a read may change bit status.

This register contains the data to be transmitted when in the Slave Transmit state, and the data received when in the Slave Receive state.

I2C Slave Data (I2CSDR)

I2C Slave 0 base: 0x4002.0800
I2C Slave 1 base: 0x4002.1800
Offset 0x008
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DATA							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x00	Data for Transfer This field contains the data for transfer during a slave receive or transmit operation.

Register 13: I^2C Slave Interrupt Mask (I2CSIMR), offset 0x00C

This register controls whether a raw interrupt is promoted to a controller interrupt.

I2C Slave Interrupt Mask (I2CSIMR)

I2C Slave 0 base: 0x4002.0800
I2C Slave 1 base: 0x4002.1800
Offset 0x00C
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													STOPIM	STARTIM	DATAIM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Value	Description
1	The STOP condition interrupt is sent to the interrupt controller when the `STOPRIS` bit in the **I2CSRIS** register is set.
0	The `STOPRIS` interrupt is suppressed and not sent to the interrupt controller.

1	STARTIM	RO	0	Start Condition Interrupt Mask

Value	Description
1	The START condition interrupt is sent to the interrupt controller when the `STARTRIS` bit in the **I2CSRIS** register is set.
0	The `STARTRIS` interrupt is suppressed and not sent to the interrupt controller.

0	DATAIM	R/W	0	Data Interrupt Mask

Value	Description
1	The data received or data requested interrupt is sent to the interrupt controller when the `DATARIS` bit in the **I2CSRIS** register is set.
0	The `DATARIS` interrupt is suppressed and not sent to the interrupt controller.

Register 14: I^{2}C Slave Raw Interrupt Status (I2CSRIS), offset 0x010

This register specifies whether an interrupt is pending.

I2C Slave Raw Interrupt Status (I2CSRIS)

I2C Slave 0 base: 0x4002.0800
I2C Slave 1 base: 0x4002.1800
Offset 0x010
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													STOPRIS	STARTRIS	DATARIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPRIS	RO	0	Stop Condition Raw Interrupt Status Value Description 1 A STOP condition interrupt is pending. 0 No interrupt. This bit is cleared by writing a 1 to the `STOPIC` bit in the **I2CSICR** register.
1	STARTRIS	RO	0	Start Condition Raw Interrupt Status Value Description 1 A START condition interrupt is pending. 0 No interrupt. This bit is cleared by writing a 1 to the `STARTIC` bit in the **I2CSICR** register.
0	DATARIS	RO	0	Data Raw Interrupt Status Value Description 1 A data received or data requested interrupt is pending. 0 No interrupt. This bit is cleared by writing a 1 to the `DATAIC` bit in the **I2CSICR** register.

Register 15: I^{2}C Slave Masked Interrupt Status (I2CSMIS), offset 0x014

This register specifies whether an interrupt was signaled.

I2C Slave Masked Interrupt Status (I2CSMIS)

I2C Slave 0 base: 0x4002.0800
I2C Slave 1 base: 0x4002.1800
Offset 0x014
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													STOPMIS	STARTMIS	DATAMIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPMIS	R/W	0	Stop Condition Masked Interrupt Status Value Description 1 An unmasked STOP condition interrupt was signaled is pending. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `STOPIC` bit in the **I2CSICR** register.
1	STARTMIS	R/W	0	Start Condition Masked Interrupt Status Value Description 1 An unmasked START condition interrupt was signaled is pending. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `STARTIC` bit in the **I2CSICR** register.
0	DATAMIS	RO	0	Data Masked Interrupt Status Value Description 1 An unmasked data received or data requested interrupt was signaled is pending. 0 An interrupt has not occurred or is masked. This bit is cleared by writing a 1 to the `DATAIC` bit in the **I2CSICR** register.

Register 16: I^2C Slave Interrupt Clear (I2CSICR), offset 0x018

This register clears the raw interrupt. A read of this register returns no meaningful data.

I2C Slave Interrupt Clear (I2CSICR)

I2C Slave 0 base: 0x4002.0800
I2C Slave 1 base: 0x4002.1800
Offset 0x018
Type WO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													STOPIC	STARTIC	DATAIC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPIC	WO	0	Stop Condition Interrupt Clear Writing a 1 to this bit clears the `STOPRIS` bit in the **I2CSRIS** register and the `STOPMIS` bit in the **I2CSMIS** register. A read of this register returns no meaningful data.
1	STARTIC	WO	0	Start Condition Interrupt Clear Writing a 1 to this bit clears the `STOPRIS` bit in the **I2CSRIS** register and the `STOPMIS` bit in the **I2CSMIS** register. A read of this register returns no meaningful data.
0	DATAIC	WO	0	Data Interrupt Clear Writing a 1 to this bit clears the `STOPRIS` bit in the **I2CSRIS** register and the `STOPMIS` bit in the **I2CSMIS** register. A read of this register returns no meaningful data.

17 Analog Comparators

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

Note: Not all comparators have the option to drive an output pin.

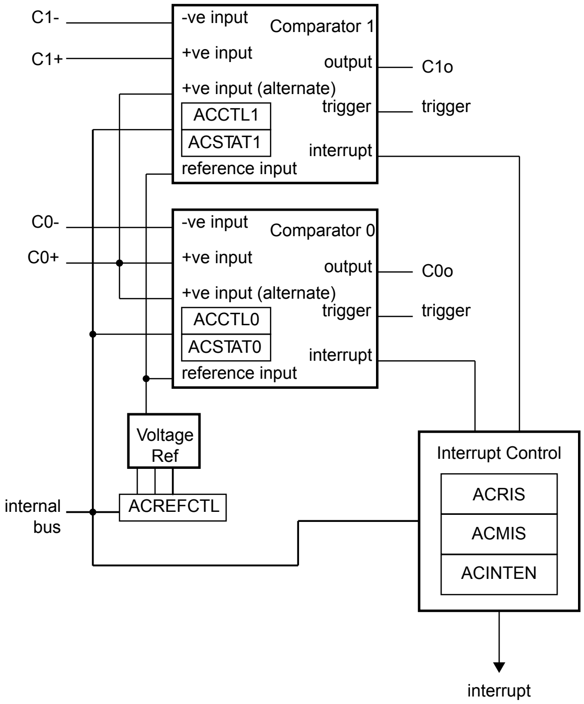
The comparator can provide its output to a device pin, acting as a replacement for an analog comparator on the board. In addition, the comparator can signal the application via interrupts or trigger the start of a sample sequence in the ADC. The interrupt generation and ADC triggering logic is separate and independent. This flexibility means, for example, that an interrupt can be generated on a rising edge and the ADC triggered on a falling edge.

The Stellaris® LM3S1W16 microcontroller provides two independent integrated analog comparators with the following functions:

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

17.1 Block Diagram

Figure 17-1. Analog Comparator Module Block Diagram



17.2 Signal Description

Table 17-1 on page 615 lists the external signals of the Analog Comparators and describes the function of each. The Analog Comparator output signals are alternate functions for some GPIO signals and default to be GPIO signals at reset. The column in the table below titled "Pin Mux/Pin Assignment" lists the possible GPIO pin placements for the Analog Comparator signals. The `AFSEL` bit in the **GPIO Alternate Function Select (GPIOAFSEL)** register (page 307) should be set to choose the Analog Comparator function. The number in parentheses is the encoding that must be programmed into the `PMCn` field in the **GPIO Port Control (GPIOPCTL)** register (page 324) to assign the Analog Comparator signal to the specified GPIO port pin. The positive and negative input signals are configured by clearing the `DEN` bit in the **GPIO Digital Enable (GPIODEN)** register. For more information on configuring GPIOs, see “General-Purpose Input/Outputs (GPIOs)” on page 285.

Table 17-1. Signals for Analog Comparators

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`C0+`	56	PB6	I	Analog	Analog comparator 0 positive input.
`C0-`	58	PB4	I	Analog	Analog comparator 0 negative input.
`C0o`	14 56 57	PC5 (3) PB6 (3) PB5 (1)	O	TTL	Analog comparator 0 output.
`C1+`	14	PC5	I	Analog	Analog comparator 1 positive input.
`C1-`	57	PB5	I	Analog	Analog comparator 1 negative input.
`C1o`	14 16	PC5 (2) PC7 (7)	O	TTL	Analog comparator 1 output.

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

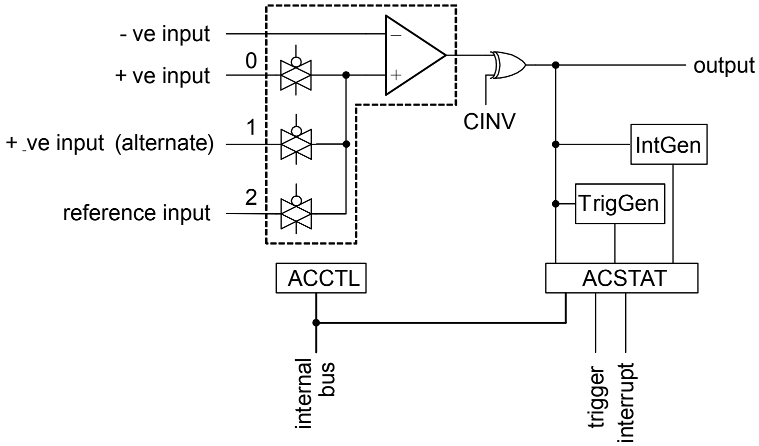
17.3 Functional Description

The comparator compares the VIN- and VIN+ inputs to produce an output, VOUT.

```
VIN- < VIN+, VOUT = 1
VIN- > VIN+, VOUT = 0
```

As shown in Figure 17-2 on page 616, the input source for VIN- is an external input, `Cn-`. In addition to an external input, `Cn+`, input sources for VIN+ can be the C0+ or an internal reference, V_{IREF}.

Figure 17-2. Structure of Comparator Unit



A comparator is configured through two status/control registers, **Analog Comparator Control (ACCTL)** and **Analog Comparator Status (ACSTAT)**. The internal reference is configured through one control register, **Analog Comparator Reference Voltage Control (ACREFCTL)**. Interrupt status and control are configured through three registers, **Analog Comparator Masked Interrupt Status (ACMIS)**, **Analog Comparator Raw Interrupt Status (ACRIS)**, and **Analog Comparator Interrupt Enable (ACINTEN)**.

Typically, the comparator output is used internally to generate an interrupt as controlled by the `ISEN` bit in the **ACCTL** register. The output may also be used to drive an external pin, Co or generate an analog-to-digital converter (ADC) trigger.

Important: The ASRCP bits in the **ACCTL** register must be set before using the analog comparators.

17.3.1 Internal Reference Programming

The structure of the internal reference is shown in Figure 17-3 on page 616. The internal reference is controlled by a single configuration register (**ACREFCTL**). Table 17-2 on page 617 shows the programming options to develop specific internal reference values, to compare an external voltage against a particular voltage generated internally (V_{IREF}).

Figure 17-3. Comparator Internal Reference Structure

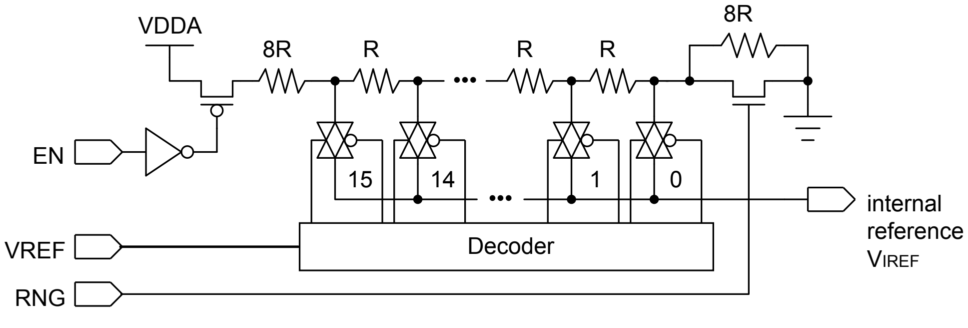


Table 17-2. Internal Reference Voltage and ACREFCTL Field Values

ACREFCTL Register		Output Reference Voltage Based on VREF Field Value
`EN` Bit Value	`RNG` Bit Value	
`EN=0`	`RNG=X`	0 V (GND) for any value of `VREF`; however, it is recommended that `RNG=1` and `VREF=0` for the least noisy ground reference.
`EN=1`	`RNG=0`	Total resistance in ladder is 31 R. $V_{IREF} = V_{DDA} \times \frac{R_{VREF}}{R_T}$ $V_{IREF} = V_{DDA} \times \frac{(VREF + 8)}{31}$ $V_{IREF} = 0.85 + 0.106 \times VREF$ The range of internal reference in this mode is 0.85-2.448 V.
	`RNG=1`	Total resistance in ladder is 23 R. $V_{IREF} = V_{DDA} \times \frac{R_{VREF}}{R_T}$ $V_{IREF} = V_{DDA} \times \frac{VREF}{23}$ $V_{IREF} = 0.143 \times VREF$ The range of internal reference for this mode is 0-2.152 V.

17.4 Initialization and Configuration

The following example shows how to configure an analog comparator to read back its output value from an internal register.

1. Enable the analog comparator 0 clock by writing a value of 0x0010.0000 to the **RCGC1** register in the System Control module (see page 147).

2. In the GPIO module, enable the GPIO port/pin associated with the input signals as GPIO inputs. To determine which GPIO to configure, see Table 19-4 on page 636.

3. Configure the `PMCn` fields in the **GPIOPCTL** register to assign the analog comparator output signals to the appropriate pins (see page 324 and Table 19-5 on page 640).

4. Configure the internal voltage reference to 1.65 V by writing the **ACREFCTL** register with the value 0x0000.030C.

5. Configure the comparator to use the internal voltage reference and to *not* invert the output by writing the **ACCTLn** register with the value of 0x0000.040C.

6. Delay for 10 µs.

7. Read the comparator output value by reading the **ACSTATn** register's `OVAL` value.

Change the level of the comparator negative input signal `C-` to see the `OVAL` value change.

17.5 Register Map

Table 17-3 on page 618 lists the comparator registers. The offset listed is a hexadecimal increment to the register's address, relative to the Analog Comparator base address of 0x4003.C000. Note that the analog comparator clock must be enabled before the registers can be programmed (see page 147).

Table 17-3. Analog Comparators Register Map

Offset	Name	Type	Reset	Description	See page
0x000	ACMIS	R/W1C	0x0000.0000	Analog Comparator Masked Interrupt Status	619
0x004	ACRIS	RO	0x0000.0000	Analog Comparator Raw Interrupt Status	620
0x008	ACINTEN	R/W	0x0000.0000	Analog Comparator Interrupt Enable	621
0x010	ACREFCTL	R/W	0x0000.0000	Analog Comparator Reference Voltage Control	622
0x020	ACSTAT0	RO	0x0000.0000	Analog Comparator Status 0	623
0x024	ACCTL0	R/W	0x0000.0000	Analog Comparator Control 0	624
0x040	ACSTAT1	RO	0x0000.0000	Analog Comparator Status 1	623
0x044	ACCTL1	R/W	0x0000.0000	Analog Comparator Control 1	624

17.6 Register Descriptions

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

Register 1: Analog Comparator Masked Interrupt Status (ACMIS), offset 0x000

This register provides a summary of the interrupt status (masked) of the comparator.

Analog Comparator Masked Interrupt Status (ACMIS)

Base 0x4003.C000
Offset 0x000
Type R/W1C, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	IN1	R/W1C	0	Comparator 1 Masked Interrupt Status Value Description 1 The `IN1` bits in the **ACRIS** register and the **ACINTEN** registers are set, providing an interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the `IN1` bit in the **ACRIS** register.
0	IN0	R/W1C	0	Comparator 0 Masked Interrupt Status Value Description 1 The `IN0` bits in the **ACRIS** register and the **ACINTEN** registers are set, providing an interrupt to the interrupt controller. 0 No interrupt has occurred or the interrupt is masked. This bit is cleared by writing a 1. Clearing this bit also clears the `IN0` bit in the **ACRIS** register.

Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004

This register provides a summary of the interrupt status (raw) of the comparator. The bits in this register must be enabled to generate interrupts using the **ACINTEN** register.

Analog Comparator Raw Interrupt Status (ACRIS)

Base 0x4003.C000
Offset 0x004
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														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
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	IN1	RO	0	Comparator 1 Interrupt Status Value Description 1 Comparator 1 has generated an interruptfor an event as configured by the `ISEN` bit in the **ACCTL1** register. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the `IN1` bit in the **ACMIS** register.
0	IN0	RO	0	Comparator 0 Interrupt Status Value Description 1 Comparator 0 has generated an interrupt for an event as configured by the `ISEN` bit in the **ACCTL0** register. 0 An interrupt has not occurred. This bit is cleared by writing a 1 to the `IN0` bit in the **ACMIS** register.

Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x008

This register provides the interrupt enable for the comparators.

Analog Comparator Interrupt Enable (ACINTEN)

Base 0x4003.C000
Offset 0x008
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	IN1	R/W	0	Comparator 1 Interrupt Enable Value Description 1 The raw interrupt signal comparator 1 is sent to the interrupt controller. 0 A comparator 1 interrupt does not affect the interrupt status.
0	IN0	R/W	0	Comparator 0 Interrupt Enable Value Description 1 The raw interrupt signal comparator 0 is sent to the interrupt controller. 0 A comparator 0 interrupt does not affect the interrupt status.

Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x010

This register specifies whether the resistor ladder is powered on as well as the range and tap.

Analog Comparator Reference Voltage Control (ACREFCTL)

Base 0x4003.C000
Offset 0x010
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	EN	R/W	0	Resistor Ladder Enable Value Description 0 The resistor ladder is unpowered. 1 Powers on the resistor ladder. The resistor ladder is connected to V_{DDA}. This bit is cleared at reset so that the internal reference consumes the least amount of power if it is not used.
8	RNG	R/W	0	Resistor Ladder Range Value Description 0 The resistor ladder has a total resistance of 31 R. 1 The resistor ladder has a total resistance of 23 R.
7:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	VREF	R/W	0x0	Resistor Ladder Voltage Ref The VREF bit field specifies the resistor ladder tap that is passed through an analog multiplexer. The voltage corresponding to the tap position is the internal reference voltage available for comparison. See Table 17-2 on page 617 for some output reference voltage examples.

Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x020

Register 6: Analog Comparator Status 1 (ACSTAT1), offset 0x040

These registers specify the current output value of the comparator.

Analog Comparator Status 0 (ACSTAT0)

Base 0x4003.C000
Offset 0x020
Type RO, reset 0x0000.0000

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

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	OVAL	RO	0	Comparator Output Value Value Description 0 VIN- > VIN+ 1 VIN- < VIN+ VIN - is the voltage on the `Cn-` pin. VIN+ is the voltage on the `Cn+` pin, the `C0+` pin, or the internal voltage reference (V_{IREF}) as defined by the `ASRCP` bit in the **ACCTL** register.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 7: Analog Comparator Control 0 (ACCTL0), offset 0x024

Register 8: Analog Comparator Control 1 (ACCTL1), offset 0x044

These registers configure the comparator's input and output.

Analog Comparator Control 0 (ACCTL0)

Base 0x4003.C000
Offset 0x024
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				TOEN	ASRCP		reserved	TSLVAL	TSEN		ISLVAL	ISEN		CINV	reserved
Type	RO	RO	RO	RO	R/W	R/W	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	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TOEN	R/W	0	Trigger Output Enable Value Description 0 ADC events are suppressed and not sent to the ADC. 1 ADC events are sent to the ADC.
10:9	ASRCP	R/W	0x0	Analog Source Positive The `ASRCP` field specifies the source of input voltage to the VIN+ terminal of the comparator. The encodings for this field are as follows: Value Description 0x0 Pin value of `Cn+` 0x1 Pin value of `C0+` 0x2 Internal voltage reference (V_{IREF}) 0x3 Reserved
8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TSLVAL	R/W	0	Trigger Sense Level Value Value Description 0 An ADC event is generated if the comparator output is Low. 1 An ADC event is generated if the comparator output is High.

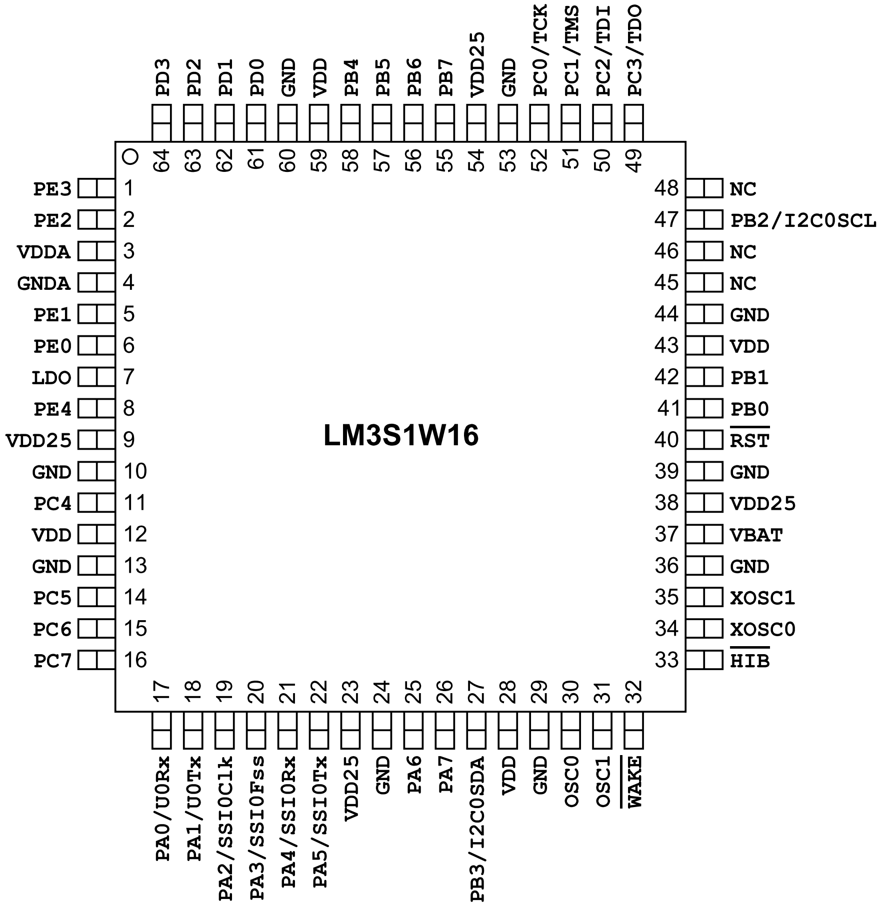
Bit/Field	Name	Type	Reset	Description
6:5	TSEN	R/W	0x0	Trigger Sense The `TSEN` field specifies the sense of the comparator output that generates an ADC event. The sense conditioning is as follows: Value Description 0x0 Level sense, see `TSLVAL` 0x1 Falling edge 0x2 Rising edge 0x3 Either edge
4	ISLVAL	R/W	0	Interrupt Sense Level Value Value Description 0 An interrupt is generated if the comparator output is Low. 1 An interrupt is generated if the comparator output is High.
3:2	ISEN	R/W	0x0	Interrupt Sense The `ISEN` field specifies the sense of the comparator output that generates an interrupt. The sense conditioning is as follows: Value Description 0x0 Level sense, see `ISLVAL` 0x1 Falling edge 0x2 Rising edge 0x3 Either edge
1	CINV	R/W	0	Comparator Output Invert Value Description 0 The output of the comparator is unchanged. 1 The output of the comparator is inverted prior to being processed by hardware.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

18 Pin Diagram

The LM3S1W16 microcontroller pin diagram is shown below.

Each GPIO signal is identified by its GPIO port unless it defaults to an alternate function on reset. In this case, the GPIO port name is followed by the default alternate function. To see a complete list of possible functions for each pin, see Table 19-5 on page 640.

Figure 18-1. 64-Pin LQFP Package Pin Diagram



19 Signal Tables

The following tables list the signals available for each pin. Signals are configured as GPIOs on reset, except for those noted below. Use the **GPIOAMSEL** register (see page 323) to select analog mode. For a GPIO pin to be used for an alternate digital function, the corresponding bit in the **GPIOAFSEL** register (see page 307) must be set. Further pin muxing options are provided through the `PMCx` bit field in the **GPIOPCTL** register (see page 324), which selects one of several available peripheral functions for that GPIO.

Important: All GPIO pins are configured as GPIOs by default with the exception of the pins shown in the table below. A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ puts the pins back to their default state.

Table 19-1. GPIO Pins With Default Alternate Functions

GPIO Pin	Default State	GPIOAFSEL Bit	GPIOPCTL PMCx Bit Field
PA[1:0]	UART0	1	0x1
PA[5:2]	SSI0	1	0x1
PB[3:2]	I^2C0	1	0x1
PC[3:0]	JTAG/SWD	1	0x3

Table 19-2 on page 627 shows the pin-to-signal-name mapping, including functional characteristics of the signals. Each possible alternate analog and digital function is listed for each pin.

Table 19-3 on page 632 lists the signals in alphabetical order by signal name. If it is possible for a signal to be on multiple pins, each possible pin assignment is listed. The "Pin Mux" column indicates the GPIO and the encoding needed in the `PMCx` bit field in the **GPIOPCTL** register.

Table 19-4 on page 636 groups the signals by functionality, except for GPIOs. If it is possible for a signal to be on multiple pins, each possible pin assignment is listed.

Table 19-5 on page 640 lists the GPIO pins and their analog and digital alternate functions. The `AINx` and `VREFA` analog signals are not 5-V tolerant and go through an isolation circuit before reaching their circuitry. These signals are configured by clearing the corresponding `DEN` bit in the **GPIO Digital Enable (GPIODEN)** register and setting the corresponding `AMSEL` bit in the **GPIO Analog Mode Select (GPIOAMSEL)** register. Other analog signals are 5-V tolerant and are connected directly to their circuitry (`C0-`, `C0+`, `C1-`, `C1+`). These signals are configured by clearing the `DEN` bit in the **GPIO Digital Enable (GPIODEN)** register. The digital signals are enabled by setting the appropriate bit in the **GPIO Alternate Function Select (GPIOAFSEL)** and **GPIODEN** registers and configuring the `PMCx` bit field in the **GPIO Port Control (GPIOPCTL)** register to the numeric enoding shown in the table below. Table entries that are shaded gray are the default values for the corresponding GPIO pin.

Table 19-2. Signals by Pin Number

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
1	`PE3`	I/O	TTL	GPIO port E bit 3.
	`AIN0`	I	Analog	Analog-to-digital converter input 0.
	`CCP1`	I/O	TTL	Capture/Compare/PWM 1.
	`SSI1Tx`	O	TTL	SSI module 1 transmit.

Table 19-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
2	`PE2`	I/O	TTL	GPIO port E bit 2.
	`AIN1`	I	Analog	Analog-to-digital converter input 1.
	`CCP4`	I/O	TTL	Capture/Compare/PWM 4.
	`SSI1Rx`	I	TTL	SSI module 1 receive.
	`CCP2`	I/O	TTL	Capture/Compare/PWM 2.
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	`PE1`	I/O	TTL	GPIO port E bit 1.
	`SSI1Fss`	I/O	TTL	SSI module 1 frame.
	`CCP2`	I/O	TTL	Capture/Compare/PWM 2.
	`AIN2`	I	Analog	Analog-to-digital converter input 2.
6	`PE0`	I/O	TTL	GPIO port E bit 0.
	`SSI1Clk`	I/O	TTL	SSI module 1 clock.
	`CCP3`	I/O	TTL	Capture/Compare/PWM 3.
	`AIN3`	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 µF or greater. When the on-chip LDO is used to provide power to the logic, the `LDO` pin must also be connected to the `VDDC` pins at the board level in addition to the decoupling capacitor(s).
8	`PE4`	I/O	TTL	GPIO port E bit 4.
	`CCP3`	I/O	TTL	Capture/Compare/PWM 3.
	`U2Tx`	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
	`CCP2`	I/O	TTL	Capture/Compare/PWM 2.
9	`VDD25`	ERR:	ERR:	Positive supply for most of the logic function, including the processor core and most peripherals.
10	`GND`	-	Power	Ground reference for logic and I/O pins.
11	`PC4`	I/O	TTL	GPIO port C bit 4.
	`CCP5`	I/O	TTL	Capture/Compare/PWM 5.
	`CCP2`	I/O	TTL	Capture/Compare/PWM 2.
	`CCP4`	I/O	TTL	Capture/Compare/PWM 4.
	`CCP1`	I/O	TTL	Capture/Compare/PWM 1.
12	`VDD`	-	Power	Positive supply for I/O and some logic.
13	`GND`	-	Power	Ground reference for logic and I/O pins.

Table 19-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
14	`PC5`	I/O	TTL	GPIO port C bit 5.
	`C1+`	I	Analog	Analog comparator 1 positive input.
	`CCP1`	I/O	TTL	Capture/Compare/PWM 1.
	`C1o`	O	TTL	Analog comparator 1 output.
	`C0o`	O	TTL	Analog comparator 0 output.
	`CCP3`	I/O	TTL	Capture/Compare/PWM 3.
15	`PC6`	I/O	TTL	GPIO port C bit 6.
	`CCP3`	I/O	TTL	Capture/Compare/PWM 3.
	`U1Rx`	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	`CCP0`	I/O	TTL	Capture/Compare/PWM 0.
16	`PC7`	I/O	TTL	GPIO port C bit 7.
	`CCP4`	I/O	TTL	Capture/Compare/PWM 4.
	`CCP0`	I/O	TTL	Capture/Compare/PWM 0.
	`U1Tx`	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	`C1o`	O	TTL	Analog comparator 1 output.
17	`PA0`	I/O	TTL	GPIO port A bit 0.
	`U0Rx`	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
	`I2C1SCL`	I/O	OD	I^2C module 1 clock.
	`U1Rx`	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
18	`PA1`	I/O	TTL	GPIO port A bit 1.
	`U0Tx`	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
	`I2C1SDA`	I/O	OD	I^2C module 1 data.
	`U1Tx`	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
19	`PA2`	I/O	TTL	GPIO port A bit 2.
	`SSI0Clk`	I/O	TTL	SSI module 0 clock.
20	`PA3`	I/O	TTL	GPIO port A bit 3.
	`SSI0Fss`	I/O	TTL	SSI module 0 frame.
21	`PA4`	I/O	TTL	GPIO port A bit 4.
	`SSI0Rx`	I	TTL	SSI module 0 receive.
22	`PA5`	I/O	TTL	GPIO port A bit 5.
	`SSI0Tx`	O	TTL	SSI module 0 transmit.
23	`VDD25`	ERR:	ERR:	Positive supply for most of the logic function, including the processor core and most peripherals.
24	`GND`	-	Power	Ground reference for logic and I/O pins.
25	`PA6`	I/O	TTL	GPIO port A bit 6.
	`I2C1SCL`	I/O	OD	I^2C module 1 clock.
	`CCP1`	I/O	TTL	Capture/Compare/PWM 1.

Table 19-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
26	PA7	I/O	TTL	GPIO port A bit 7.
	I2C1SDA	I/O	OD	I^{2}C module 1 data.
	CCP4	I/O	TTL	Capture/Compare/PWM 4.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
27	PB3	I/O	TTL	GPIO port B bit 3.
	I2C0SDA	I/O	OD	I^{2}C module 0 data.
28	VDD	-	Power	Positive supply for I/O and some logic.
29	GND	-	Power	Ground reference for logic and I/O pins.
30	OSC0	I	Analog	Main oscillator crystal input or an external clock reference input.
31	OSC1	O	Analog	Main oscillator crystal output.
32	$\overline{\text{WAKE}}$	I	TTL	An external input that brings the processor out of Hibernate mode when asserted.
33	$\overline{\text{HIB}}$	O	OD	An open-drain output that indicates the processor is in Hibernate mode.
34	XOSC0	I	Analog	Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the `CLKSEL` bit in the **HIBCTL** register.
35	XOSC1	O	Analog	Hibernation module oscillator crystal output.
36	GND	-	Power	Ground reference for logic and I/O pins.
37	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.
38	VDD25	ERR:	ERR:	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	$\overline{\text{RST}}$	I	TTL	System reset input.
41	PB0	I/O	TTL	GPIO port B bit 0.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
42	PB1	I/O	TTL	GPIO port B bit 1.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
43	VDD	-	Power	Positive supply for I/O and some logic.
44	GND	-	Power	Ground reference for logic and I/O pins.
45	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
46	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
47	PB2	I/O	TTL	GPIO port B bit 2.
	I2C0SCL	I/O	OD	I^{2}C module 0 clock.
	CCP3	I/O	TTL	Capture/Compare/PWM 3.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.

Table 19-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
48	NC	-	-	No connect. Leave the pin electrically unconnected/isolated.
49	PC3	I/O	TTL	GPIO port C bit 3.
	TDO	O	TTL	JTAG TDO and SWO.
	SWO	O	TTL	JTAG TDO and SWO.
50	PC2	I/O	TTL	GPIO port C bit 2.
	TDI	I	TTL	JTAG TDI.
51	PC1	I/O	TTL	GPIO port C bit 1.
	TMS	I	TTL	JTAG TMS and SWDIO.
	SWDIO	I/O	TTL	JTAG TMS and SWDIO.
52	PC0	I/O	TTL	GPIO port C bit 0.
	TCK	I	TTL	JTAG/SWD CLK.
	SWCLK	I	TTL	JTAG/SWD CLK.
53	GND	-	Power	Ground reference for logic and I/O pins.
54	VDD25	ERR:	ERR:	Positive supply for most of the logic function, including the processor core and most peripherals.
55	PB7	I/O	TTL	GPIO port B bit 7.
	NMI	I	TTL	Non-maskable interrupt.
56	PB6	I/O	TTL	GPIO port B bit 6.
	VREFA	I	Analog	This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to VREFA is the voltage with which an AINn signal is converted to 1023. The VREFA input is limited to the range specified in Table 21-2 on page 643.
	C0+	I	Analog	Analog comparator 0 positive input.
	CCP1	I/O	TTL	Capture/Compare/PWM 1.
	C0o	O	TTL	Analog comparator 0 output.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
57	PB5	I/O	TTL	GPIO port B bit 5.
	C1-	I	Analog	Analog comparator 1 negative input.
	C0o	O	TTL	Analog comparator 0 output.
	CCP5	I/O	TTL	Capture/Compare/PWM 5.
	CCP0	I/O	TTL	Capture/Compare/PWM 0.
	CCP2	I/O	TTL	Capture/Compare/PWM 2.
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
58	PB4	I/O	TTL	GPIO port B bit 4.
	C0-	I	Analog	Analog comparator 0 negative input.
	U2Rx	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
59	VDD	-	Power	Positive supply for I/O and some logic.
60	GND	-	Power	Ground reference for logic and I/O pins.

Table 19-2. Signals by Pin Number *(continued)*

Pin Number	Pin Name	Pin Type	Buffer Type[a]	Description
61	`PD0`	I/O	TTL	GPIO port D bit 0.
	`AIN7`	I	Analog	Analog-to-digital converter input 7.
	`U2Rx`	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
	`U1Rx`	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
62	`PD1`	I/O	TTL	GPIO port D bit 1.
	`AIN6`	I	Analog	Analog-to-digital converter input 6.
	`U2Tx`	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
	`U1Tx`	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	`CCP2`	I/O	TTL	Capture/Compare/PWM 2.
63	`PD2`	I/O	TTL	GPIO port D bit 2.
	`AIN5`	I	Analog	Analog-to-digital converter input 5.
	`U1Rx`	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	`CCP5`	I/O	TTL	Capture/Compare/PWM 5.
64	`PD3`	I/O	TTL	GPIO port D bit 3.
	`AIN4`	I	Analog	Analog-to-digital converter input 4.
	`U1Tx`	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	`CCP0`	I/O	TTL	Capture/Compare/PWM 0.

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

Table 19-3. Signals by Signal Name

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`AIN0`	1	PE3	I	Analog	Analog-to-digital converter input 0.
`AIN1`	2	PE2	I	Analog	Analog-to-digital converter input 1.
`AIN2`	5	PE1	I	Analog	Analog-to-digital converter input 2.
`AIN3`	6	PE0	I	Analog	Analog-to-digital converter input 3.
`AIN4`	64	PD3	I	Analog	Analog-to-digital converter input 4.
`AIN5`	63	PD2	I	Analog	Analog-to-digital converter input 5.
`AIN6`	62	PD1	I	Analog	Analog-to-digital converter input 6.
`AIN7`	61	PD0	I	Analog	Analog-to-digital converter input 7.
`C0+`	56	PB6	I	Analog	Analog comparator 0 positive input.
`C0-`	58	PB4	I	Analog	Analog comparator 0 negative input.
`C0o`	14 56 57	PC5 (3) PB6 (3) PB5 (1)	O	TTL	Analog comparator 0 output.
`C1+`	14	PC5	I	Analog	Analog comparator 1 positive input.
`C1-`	57	PB5	I	Analog	Analog comparator 1 negative input.

Table 19-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`C1o`	14 16	PC5 (2) PC7 (7)	O	TTL	Analog comparator 1 output.
`CCP0`	15 16 41 47 57 64	PC6 (6) PC7 (4) PB0 (1) PB2 (5) PB5 (4) PD3 (4)	I/O	TTL	Capture/Compare/PWM 0.
`CCP1`	1 11 14 25 42 56	PE3 (1) PC4 (9) PC5 (1) PA6 (2) PB1 (4) PB6 (1)	I/O	TTL	Capture/Compare/PWM 1.
`CCP2`	2 5 8 11 42 57 62	PE2 (5) PE1 (4) PE4 (6) PC4 (5) PB1 (1) PB5 (6) PD1 (10)	I/O	TTL	Capture/Compare/PWM 2.
`CCP3`	6 8 14 15 26 47	PE0 (3) PE4 (1) PC5 (5) PC6 (1) PA7 (7) PB2 (4)	I/O	TTL	Capture/Compare/PWM 3.
`CCP4`	2 11 16 26	PE2 (1) PC4 (6) PC7 (1) PA7 (2)	I/O	TTL	Capture/Compare/PWM 4.
`CCP5`	11 56 57 63	PC4 (1) PB6 (6) PB5 (2) PD2 (4)	I/O	TTL	Capture/Compare/PWM 5.
`GND`	10 13 24 29 36 39 44 53 60	fixed	-	Power	Ground reference for logic and I/O pins.
`GNDA`	4	fixed	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on `VDD` from affecting the analog functions.
$\overline{\mathtt{HIB}}$	33	fixed	O	OD	An open-drain output that indicates the processor is in Hibernate mode.
`I2C0SCL`	47	PB2 (1)	I/O	OD	I^{2}C module 0 clock.
`I2C0SDA`	27	PB3 (1)	I/O	OD	I^{2}C module 0 data.

Table 19-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`I2C1SCL`	17 25	PA0 (8) PA6 (1)	I/O	OD	I^{2}C module 1 clock.
`I2C1SDA`	18 26	PA1 (8) PA7 (1)	I/O	OD	I^{2}C module 1 data.
`LDO`	7	fixed	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. When the on-chip LDO is used to provide power to the logic, the `LDO` pin must also be connected to the `VDDC` pins at the board level in addition to the decoupling capacitor(s).
NC	45 46 48	fixed	-	-	No connect. Leave the pin electrically unconnected/isolated.
`NMI`	55	PB7 (4)	I	TTL	Non-maskable interrupt.
`OSC0`	30	fixed	I	Analog	Main oscillator crystal input or an external clock reference input.
`OSC1`	31	fixed	O	Analog	Main oscillator crystal output.
`PA0`	17	-	I/O	TTL	GPIO port A bit 0.
`PA1`	18	-	I/O	TTL	GPIO port A bit 1.
`PA2`	19	-	I/O	TTL	GPIO port A bit 2.
`PA3`	20	-	I/O	TTL	GPIO port A bit 3.
`PA4`	21	-	I/O	TTL	GPIO port A bit 4.
`PA5`	22	-	I/O	TTL	GPIO port A bit 5.
`PA6`	25	-	I/O	TTL	GPIO port A bit 6.
`PA7`	26	-	I/O	TTL	GPIO port A bit 7.
`PB0`	41	-	I/O	TTL	GPIO port B bit 0.
`PB1`	42	-	I/O	TTL	GPIO port B bit 1.
`PB2`	47	-	I/O	TTL	GPIO port B bit 2.
`PB3`	27	-	I/O	TTL	GPIO port B bit 3.
`PB4`	58	-	I/O	TTL	GPIO port B bit 4.
`PB5`	57	-	I/O	TTL	GPIO port B bit 5.
`PB6`	56	-	I/O	TTL	GPIO port B bit 6.
`PB7`	55	-	I/O	TTL	GPIO port B bit 7.
`PC0`	52	-	I/O	TTL	GPIO port C bit 0.
`PC1`	51	-	I/O	TTL	GPIO port C bit 1.
`PC2`	50	-	I/O	TTL	GPIO port C bit 2.
`PC3`	49	-	I/O	TTL	GPIO port C bit 3.
`PC4`	11	-	I/O	TTL	GPIO port C bit 4.
`PC5`	14	-	I/O	TTL	GPIO port C bit 5.
`PC6`	15	-	I/O	TTL	GPIO port C bit 6.
`PC7`	16	-	I/O	TTL	GPIO port C bit 7.
`PD0`	61	-	I/O	TTL	GPIO port D bit 0.
`PD1`	62	-	I/O	TTL	GPIO port D bit 1.
`PD2`	63	-	I/O	TTL	GPIO port D bit 2.

Table 19-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`PD3`	64	-	I/O	TTL	GPIO port D bit 3.
`PE0`	6	-	I/O	TTL	GPIO port E bit 0.
`PE1`	5	-	I/O	TTL	GPIO port E bit 1.
`PE2`	2	-	I/O	TTL	GPIO port E bit 2.
`PE3`	1	-	I/O	TTL	GPIO port E bit 3.
`PE4`	8	-	I/O	TTL	GPIO port E bit 4.
$\overline{\texttt{RST}}$	40	fixed	I	TTL	System reset input.
`SSI0Clk`	19	PA2 (1)	I/O	TTL	SSI module 0 clock.
`SSI0Fss`	20	PA3 (1)	I/O	TTL	SSI module 0 frame.
`SSI0Rx`	21	PA4 (1)	I	TTL	SSI module 0 receive.
`SSI0Tx`	22	PA5 (1)	O	TTL	SSI module 0 transmit.
`SSI1Clk`	6	PE0 (2)	I/O	TTL	SSI module 1 clock.
`SSI1Fss`	5	PE1 (2)	I/O	TTL	SSI module 1 frame.
`SSI1Rx`	2	PE2 (2)	I	TTL	SSI module 1 receive.
`SSI1Tx`	1	PE3 (2)	O	TTL	SSI module 1 transmit.
`SWCLK`	52	PC0 (3)	I	TTL	JTAG/SWD CLK.
`SWDIO`	51	PC1 (3)	I/O	TTL	JTAG TMS and SWDIO.
`SWO`	49	PC3 (3)	O	TTL	JTAG TDO and SWO.
`TCK`	52	PC0 (3)	I	TTL	JTAG/SWD CLK.
`TDI`	50	PC2 (3)	I	TTL	JTAG TDI.
`TDO`	49	PC3 (3)	O	TTL	JTAG TDO and SWO.
`TMS`	51	PC1 (3)	I	TTL	JTAG TMS and SWDIO.
`U0Rx`	17	PA0 (1)	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
`U0Tx`	18	PA1 (1)	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
`U1Rx`	15 17 41 58 61 63	PC6 (5) PA0 (9) PB0 (5) PB4 (7) PD0 (5) PD2 (1)	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
`U1Tx`	16 18 42 57 62 64	PC7 (5) PA1 (9) PB1 (5) PB5 (7) PD1 (5) PD3 (1)	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
`U2Rx`	58 61	PB4 (4) PD0 (4)	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
`U2Tx`	8 62	PE4 (5) PD1 (4)	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.
`VBAT`	37	fixed	-	Power	Power source for the Hibernation module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation module power-source supply.

Table 19-3. Signals by Signal Name *(continued)*

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
`VDD`	12 28 43 59	fixed	-	Power	Positive supply for I/O and some logic.
`VDD25`	9 23 38 54	fixed	ERR:	ERR:	Positive supply for most of the logic function, including the processor core and most peripherals.
`VDDA`	3	fixed	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
`VREFA`	56	PB6	I	Analog	This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to `VREFA` is the voltage with which an `AINn` signal is converted to 1023. The `VREFA` input is limited to the range specified in Table 21-2 on page 643.
$\overline{\texttt{WAKE}}$	32	fixed	I	TTL	An external input that brings the processor out of Hibernate mode when asserted.
`XOSC0`	34	fixed	I	Analog	Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the `CLKSEL` bit in the **HIBCTL** register.
`XOSC1`	35	fixed	O	Analog	Hibernation module oscillator crystal output.

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

Table 19-4. Signals by Function, Except for GPIO

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
ADC	`AIN0`	1	I	Analog	Analog-to-digital converter input 0.
	`AIN1`	2	I	Analog	Analog-to-digital converter input 1.
	`AIN2`	5	I	Analog	Analog-to-digital converter input 2.
	`AIN3`	6	I	Analog	Analog-to-digital converter input 3.
	`AIN4`	64	I	Analog	Analog-to-digital converter input 4.
	`AIN5`	63	I	Analog	Analog-to-digital converter input 5.
	`AIN6`	62	I	Analog	Analog-to-digital converter input 6.
	`AIN7`	61	I	Analog	Analog-to-digital converter input 7.
	`VREFA`	56	I	Analog	This input provides a reference voltage used to specify the input voltage at which the ADC converts to a maximum value. In other words, the voltage that is applied to `VREFA` is the voltage with which an `AINn` signal is converted to 1023. The `VREFA` input is limited to the range specified in Table 21-2 on page 643.

Table 19-4. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
Analog Comparators	`C0+`	56	I	Analog	Analog comparator 0 positive input.
	`C0-`	58	I	Analog	Analog comparator 0 negative input.
	`C0o`	14 56 57	O	TTL	Analog comparator 0 output.
	`C1+`	14	I	Analog	Analog comparator 1 positive input.
	`C1-`	57	I	Analog	Analog comparator 1 negative input.
	`C1o`	14 16	O	TTL	Analog comparator 1 output.
General-Purpose Timers	`CCP0`	15 16 41 47 57 64	I/O	TTL	Capture/Compare/PWM 0.
	`CCP1`	1 11 14 25 42 56	I/O	TTL	Capture/Compare/PWM 1.
	`CCP2`	2 5 8 11 42 57 62	I/O	TTL	Capture/Compare/PWM 2.
	`CCP3`	6 8 14 15 26 47	I/O	TTL	Capture/Compare/PWM 3.
	`CCP4`	2 11 16 26	I/O	TTL	Capture/Compare/PWM 4.
	`CCP5`	11 56 57 63	I/O	TTL	Capture/Compare/PWM 5.

Table 19-4. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
Hibernate	$\overline{\texttt{HIB}}$	33	O	OD	An open-drain output that indicates the processor is in Hibernate mode.
	`VBAT`	37	-	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.
	$\overline{\texttt{WAKE}}$	32	I	TTL	An external input that brings the processor out of Hibernate mode when asserted.
	`XOSC0`	34	I	Analog	Hibernation module oscillator crystal input or an external clock reference input. Note that this is either a 4.194304-MHz crystal or a 32.768-kHz oscillator for the Hibernation module RTC. See the `CLKSEL` bit in the **HIBCTL** register.
	`XOSC1`	35	O	Analog	Hibernation module oscillator crystal output.
I2C	`I2C0SCL`	47	I/O	OD	I^{2}C module 0 clock.
	`I2C0SDA`	27	I/O	OD	I^{2}C module 0 data.
	`I2C1SCL`	17 25	I/O	OD	I^{2}C module 1 clock.
	`I2C1SDA`	18 26	I/O	OD	I^{2}C module 1 data.
JTAG/SWD/SWO	`SWCLK`	52	I	TTL	JTAG/SWD CLK.
	`SWDIO`	51	I/O	TTL	JTAG TMS and SWDIO.
	`SWO`	49	O	TTL	JTAG TDO and SWO.
	`TCK`	52	I	TTL	JTAG/SWD CLK.
	`TDI`	50	I	TTL	JTAG TDI.
	`TDO`	49	O	TTL	JTAG TDO and SWO.
	`TMS`	51	I	TTL	JTAG TMS and SWDIO.

Table 19-4. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
Power	GND	10 13 24 29 36 39 44 53 60	-	Power	Ground reference for logic and I/O pins.
	GNDA	4	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
	LDO	7	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. When the on-chip LDO is used to provide power to the logic, the LDO pin must also be connected to the VDDC pins at the board level in addition to the decoupling capacitor(s).
	VDD	12 28 43 59	-	Power	Positive supply for I/O and some logic.
	VDD25	9 23 38 54	ERR:	ERR:	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.
SSI	SSI0Clk	19	I/O	TTL	SSI module 0 clock.
	SSI0Fss	20	I/O	TTL	SSI module 0 frame.
	SSI0Rx	21	I	TTL	SSI module 0 receive.
	SSI0Tx	22	O	TTL	SSI module 0 transmit.
	SSI1Clk	6	I/O	TTL	SSI module 1 clock.
	SSI1Fss	5	I/O	TTL	SSI module 1 frame.
	SSI1Rx	2	I	TTL	SSI module 1 receive.
	SSI1Tx	1	O	TTL	SSI module 1 transmit.
System Control & Clocks	NMI	55	I	TTL	Non-maskable interrupt.
	OSC0	30	I	Analog	Main oscillator crystal input or an external clock reference input.
	OSC1	31	O	Analog	Main oscillator crystal output.
	$\overline{\text{RST}}$	40	I	TTL	System reset input.

Table 19-4. Signals by Function, Except for GPIO *(continued)*

Function	Pin Name	Pin Number	Pin Type	Buffer Type[a]	Description
UART	U0Rx	17	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
	U0Tx	18	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
	U1Rx	15 17 41 58 61 63	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U1Tx	16 18 42 57 62 64	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
	U2Rx	58 61	I	TTL	UART module 2 receive. When in IrDA mode, this signal has IrDA modulation.
	U2Tx	8 62	O	TTL	UART module 2 transmit. When in IrDA mode, this signal has IrDA modulation.

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

Table 19-5. GPIO Pins and Alternate Functions

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	10	11
PA0	17	-	U0Rx	-	-	-	-	-	-	I2C1SCL	U1Rx	-	-
PA1	18	-	U0Tx	-	-	-	-	-	-	I2C1SDA	U1Tx	-	-
PA2	19	-	SSI0Clk	-	-	-	-	-	-	-	-	-	-
PA3	20	-	SSI0Fss	-	-	-	-	-	-	-	-	-	-
PA4	21	-	SSI0Rx	-	-	-	-	-	-	-	-	-	-
PA5	22	-	SSI0Tx	-	-	-	-	-	-	-	-	-	-
PA6	25	-	I2C1SCL	CCP1	-	-	-	-	-	-	-	-	-
PA7	26	-	I2C1SDA	CCP4	-	-	-	-	CCP3	-	-	-	-
PB0	41	-	CCP0	-	-	-	U1Rx	-	-	-	-	-	-
PB1	42	-	CCP2	-	-	CCP1	U1Tx	-	-	-	-	-	-
PB2	47	-	I2C0SCL	-	-	CCP3	CCP0	-	-	-	-	-	-
PB3	27	-	I2C0SDA	-	-	-	-	-	-	-	-	-	-
PB4	58	C0-	-	-	-	U2Rx	-	-	U1Rx	-	-	-	-
PB5	57	C1-	C0o	CCP5	-	CCP0	-	CCP2	U1Tx	-	-	-	-
PB6	56	VREFA C0+	CCP1	-	C0o	-	-	CCP5	-	-	-	-	-
PB7	55	-	-	-	-	NMI	-	-	-	-	-	-	-
PC0	52	-	-	-	TCK SWCLK	-	-	-	-	-	-	-	-
PC1	51	-	-	-	TMS SWDIO	-	-	-	-	-	-	-	-
PC2	50	-	-	-	TDI	-	-	-	-	-	-	-	-

Table 19-5. GPIO Pins and Alternate Functions *(continued)*

IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	10	11
PC3	49	-	-	-	TDO SWO	-	-	-	-	-	-	-	-
PC4	11	-	CCP5	-	-	-	CCP2	CCP4	-	-	CCP1	-	-
PC5	14	C1+	CCP1	C1o	C0o	-	CCP3	-	-	-	-	-	-
PC6	15	-	CCP3	-	-	-	U1Rx	CCP0	-	-	-	-	-
PC7	16	-	CCP4	-	-	CCP0	U1Tx	-	C1o	-	-	-	-
PD0	61	AIN7	-	-	-	U2Rx	U1Rx	-	-	-	-	-	-
PD1	62	AIN6	-	-	-	U2Tx	U1Tx	-	-	-	-	CCP2	-
PD2	63	AIN5	U1Rx	-	-	CCP5	-	-	-	-	-	-	-
PD3	64	AIN4	U1Tx	-	-	CCP0	-	-	-	-	-	-	-
PE0	6	AIN3	-	SSI1Clk	CCP3	-	-	-	-	-	-	-	-
PE1	5	AIN2	-	SSI1Fss	-	CCP2	-	-	-	-	-	-	-
PE2	2	AIN1	CCP4	SSI1Rx	-	-	CCP2	-	-	-	-	-	-
PE3	1	AIN0	CCP1	SSI1Tx	-	-	-	-	-	-	-	-	-
PE4	8	-	CCP3	-	-	-	U2Tx	CCP2	-	-	-	-	-

a. The digital signals that are shaded gray are the power-on default values for the corresponding GPIO pin.

20 Operating Characteristics

Table 20-1. Temperature Characteristics

Characteristic[a]	Symbol	Value	Unit
Industrial operating temperature range	T_A	-40 to +85	°C

[a] Maximum storage temperature is 150°C.

Table 20-2. Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance (junction to ambient)[a]	Θ_{JA}		°C/W
Average junction temperature[b]	T_J	$T_A + (P_{AVG} \bullet \Theta_{JA})$	°C

[a] Junction to ambient thermal resistance θ_{JA} numbers are determined by a package simulator.

[b] Power dissipation is a function of temperature.

21 Electrical Characteristics

21.1 DC Characteristics

21.1.1 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 21-1. Maximum Ratings

Parameter	Parameter Name[a]	Value		Unit
		Min	Max	
V_{DD}	I/O supply voltage (V_{DD})	0	4	V
V_{DDA}	Analog supply voltage (V_{DDA})	0	4	V
V_{BAT}	Battery supply voltage (V_{BAT})	0	4	V
V_{IN}	Input voltage	-0.3	5.5	V
I	Maximum current per output pins	-	25	mA

[a] Voltages are measured with respect to GND.

Important: This device contains circuitry to protect the inputs against damage due to high-static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either `GND` or V_{DD}).

21.1.2 Recommended DC Operating Conditions

For special high-current applications, the GPIO output buffers may be used with the following restrictions. With the GPIO pins configured as 8-mA output drivers, a total of four GPIO outputs may be used to sink current loads up to 18 mA each. At 18-mA sink current loading, the V_{OL} value is specified as 1.2 V. The high-current GPIO package pins must be selected such that there are only a maximum of two per side of the physical package with the total number of high-current GPIO outputs not exceeding four for the entire package.

Table 21-2. Recommended DC Operating Conditions

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{DD}	I/O supply voltage	3.0	3.3	3.6	V
V_{DDA}	Analog supply voltage	3.0	3.3	3.6	V
V_{DDC}[a]	Core supply voltage	1.08	1.2	1.32	V
V_{IH}	High-level input voltage	2.0	-	5.0	V
V_{IL}	Low-level input voltage	-0.3	-	1.3	V
V_{OH}[b]	High-level output voltage	2.4	-	-	V
V_{OL}[a]	Low-level output voltage	-	-	0.4	V

Table 21-2. Recommended DC Operating Conditions *(continued)*

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

a. V_{DDC} is supplied from the output of the LDO.

b. V_{OL} and V_{OH} shift to 1.2 V when using high-current GPIOs.

21.1.3 On-Chip Low Drop-Out (LDO) Regulator Characteristics

Table 21-3. LDO Regulator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
C_{LDO}	External filter capacitor size for internal power supply	1.0	-	3.0	µF
V_{LDO}	LDO output voltage	1.08	1.2	1.32	V

21.1.4 Hibernation Module Characteristics

Table 21-4. Hibernation Module DC Characteristics

Parameter	Parameter Name	Min	Nominal	Max	Unit
V_{BAT}	Battery supply voltage	2.4	3.0	3.6	V
V_{LOWBAT}	Low battery detect voltage	-	2.35	-	V

21.1.5 Flash Memory Characteristics

Table 21-5. Flash Memory Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
PE_{CYC}	Number of guaranteed mass program/erase cycles before failure[a]	15,000	-	-	cycles
T_{RET}	Data retention at average operating temperature of 125˚C	10	-	-	years
T_{PROG}	Word program time	-	-	1	ms
T_{BPROG}	Buffer program time	-	-	1	ms
T_{ERASE}	Page erase time	-	-	12	ms
T_{ME}	Mass erase time	-	-	12	ms

a. A program/erase cycle is defined as switching the bits from 1-> 0 -> 1. Caution should be used when performing block erases, as repeated block erases can shorten the number of guaranteed erase cycles, see “Flash Memory Programming” on page 196.

21.1.6 GPIO Module Characteristics

Table 21-6. GPIO Module DC Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
R_{GPIOPU}	GPIO internal pull-up resistor	50	-	110	kΩ
R_{GPIOPD}	GPIO internal pull-down resistor	55	-	180	kΩ

21.1.7 Current Specifications

This section provides information on typical and maximum power consumption under various conditions.

21.1.7.1 Preliminary Current Consumption Specifications

The following table provides preliminary figures for current consumption while ongoing characterization is completed.

Table 21-7. Preliminary Current Consumption

Parameter	Parameter Name	Conditions	Nom	Max	Unit
I_{DD_RUN}	Run mode 1 (Flash loop)	V_{DD} = 3.3 V Code= while(1){} executed in Flash Peripherals = All ON System Clock = 50 MHz (with PLL) Temp = 25°C	56	-	mA
I_{DD_SLEEP}	Sleep mode	V_{DD} = 3.3 V Peripherals = All clock gated System Clock = 50 MHz (with PLL) Temp = 25°C	8	-	mA
$I_{DD\_DEEPSLEEP}$	Deep-sleep mode	Peripherals = All OFF System Clock = IOSC30KHZ/64 Temp = 25°C	-	550	µA
I_{HIB_NORTC}	Hibernate mode (external wake, RTC disabled, I/O not powered[a])	V_{BAT} = 3.0 V V_{DD} = 0 V V_{DDA} = 0 V Peripherals = All OFF System Clock = OFF Hibernate Module = 0 kHz	8	-	µA
I_{HIB_RTC}	Hibernate mode (RTC enabled, I/O not powered[a])	V_{BAT} = 3.0 V V_{DD} = 3.3 V V_{DDA} = 3.3 V Peripherals = All OFF System Clock = OFF Hibernate Module = 32 kHz	18	-	µA

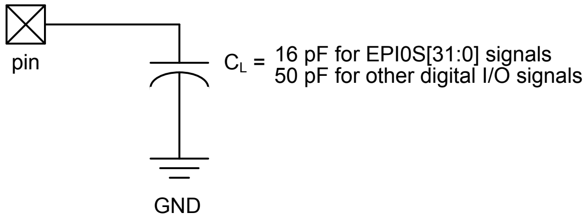
[a] The VDD3ON mode must be disabled for the I/O ring to be unpowered.

21.2 AC Characteristics

21.2.1 Load Conditions

Unless otherwise specified, the following conditions are true for all timing measurements.

Figure 21-1. Load Conditions



21.2.2 Clocks

The following sections provide specifications on the various clock sources and mode.

21.2.2.1 PLL Specifications

The following tables provide specifications for using the PLL.

Table 21-8. Phase Locked Loop (PLL) Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
f_{REF_XTAL}	Crystal reference[a]	3.579545	-	16.384	MHz
f_{REF_EXT}	External clock reference[a]	3.579545	-	16.384	MHz
f_{PLL}	PLL frequency[b]	-	400	-	MHz
T_{READY}	PLL lock time	0.562[c]	-	1.38[d]	ms

a. The exact value is determined by the crystal value programmed into the XTAL field of the **Run-Mode Clock Configuration (RCC)** register.

b. PLL frequency is automatically calculated by the hardware based on the XTAL field of the **RCC** register.

c. Using a 16.384-MHz crystal

d. Using 3.5795-MHz crystal

Table 21-9 on page 646 shows the actual frequency of the PLL based on the crystal frequency used (defined by the XTAL field in the **RCC** register).

Table 21-9. Actual PLL Frequency

XTAL	Crystal Frequency (MHz)	PLL Frequency (MHz)	Error
0x04	3.5795	400.904	0.0023%
0x05	3.6864	398.1312	0.0047%
0x06	4.0	400	-
0x07	4.096	401.408	0.0035%
0x08	4.9152	398.1312	0.0047%
0x09	5.0	400	-
0x0A	5.12	399.36	0.0016%
0x0B	6.0	400	-
0x0C	6.144	399.36	0.0016%

Table 21-9. Actual PLL Frequency *(continued)*

XTAL	Crystal Frequency (MHz)	PLL Frequency (MHz)	Error
0x0D	7.3728	398.1312	0.0047%
0x0E	8.0	400	0.0047%
0x0F	8.192	398.6773333	0.0033%
0x10	10.0	400	-
0x11	12.0	400	-
0x12	12.288	401.408	0.0035%
0x13	13.56	397.76	0.0056%
0x14	14.318	400.90904	0.0023%
0x15	16.0	400	-
0x16	16.384	404.1386667	0.010%

21.2.2.2 PIOSC Specifications

Table 21-10. PIOSC Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$f_{PIOSC25}$	Internal 16-MHz precision oscillator frequency variance, factory calibrated at 25 °C	-	±0.25%	±1%	-
f_{PIOSCT}	Internal 16-MHz precision oscillator frequency variance, factory calibrated at 25 °C, across specified temperature range	-	-	±3%	-
$f_{PIOSCUCAL}$	Internal 16-MHz precision oscillator frequency variance, user calibrated at a chosen temperature	-	±0.25%	±1%	-

21.2.2.3 Internal 30-kHz Oscillator Specifications

Table 21-11. 30-kHz Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$f_{IOSC30KHZ}$	Internal 30-KHz oscillator frequency	15	30	45	KHz

21.2.2.4 Hibernation Clock Source Specifications

Table 21-12. Hibernation Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
f_{HIBOSC}	Hibernation module oscillator frequency	-	4.194304	-	MHz
f_{HIBOSC_XTAL}	Crystal reference for hibernation oscillator	-	4.194304	-	MHz
f_{HIBOSC_EXT}	External clock reference for hibernation module	-	32.768	-	KHz
t_{HIBOSC_SETTLE}	Hibernation oscillator settling time[a]	-	-	10	ms

[a] This parameter is highly sensitive to PCB layout and trace lengths, which may make this parameter time longer. Care must be taken in PCB design to minimize trace lengths and RLC (resistance, inductance, capacitance).

Table 21-13. HIB Oscillator Input Characteristics

Name	Value	Condition
Frequency	4.194304	MHz

Table 21-13. HIB Oscillator Input Characteristics *(continued)*

Name	Value	Condition
Frequency tolerance	±100	PPM
Oscillation mode	parallel	-
Equivalent series resistance (max)	200	Ω
Load capacitance	16	pF
Drive level (typ)	100	µw

21.2.2.5 Main Oscillator Specifications

Table 21-14. Main Oscillator Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
f_{MOSC}	Main oscillator frequency	1	-	16.384	MHz
t_{MOSC_PER}	Main oscillator period	61	-	1000	ns
t_{MOSC_SETTLE}	Main oscillator settling time	17.5	-	20	ms
$f_{REF\_XTAL\_BYPASS}$	Crystal reference using the main oscillator (PLL in BYPASS mode)[a]	1	-	16.384	MHz
$f_{REF\_EXT\_BYPASS}$	External clock reference (PLL in BYPASS mode)[a]	0	-	50	MHz

a. The ADC must be clocked from the PLL or directly from a 16-MHz clock source to operate properly.

Table 21-15. MOSC Oscillator Input Characteristics

Name	Value						Condition
Frequency	16	12	8	6	4	3.5	MHz
Frequency tolerance	±100	±100	±100	±100	±100	±100	PPM
Oscillation mode	parallel	parallel	parallel	parallel	parallel	parallel	-
Equivalent series resistance (max)	70	90	120	160	200	220	Ω
Load capacitance	16	16	16	16	16	16	pF
Drive level (typ)	100	100	100	100	100	100	µw

21.2.2.6 System Clock Specifications with ADC Operation

Table 21-16. System Clock Characteristics with ADC Operation

Parameter	Parameter Name	Min	Nom	Max	Unit
f_{sysadc}	System clock frequency when the ADC module is operating (when PLL is bypassed)	16	-	-	MHz

21.2.3 JTAG and Boundary Scan

Table 21-17. 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

Table 21-17. JTAG Characteristics *(continued)*

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J3	t_{TCK_LOW}	TCK clock Low time	-	t_{TCK}	-	ns
J4	t_{TCK_HIGH}	TCK clock High time	-	t_{TCK}	-	ns
J5	t_{TCK_R}	TCK rise time	0	-	10	ns
J6	t_{TCK_F}	TCK fall time	0	-	10	ns
J7	t_{TMS_SU}	TMS setup time to TCK rise	20	-	-	ns
J8	t_{TMS_HLD}	TMS hold time from TCK rise	20	-	-	ns
J9	t_{TDI_SU}	TDI setup time to TCK rise	25	-	-	ns
J10	t_{TDI_HLD}	TDI hold time from TCK rise	25	-	-	ns
J11 t_{TDO_ZDV}	TCK fall to Data Valid from High-Z	2-mA drive	-	23	35	ns
		4-mA drive		15	26	ns
		8-mA drive		14	25	ns
		8-mA drive with slew rate control		18	29	ns
J12 t_{TDO_DV}	TCK fall to Data Valid from Data Valid	2-mA drive	-	21	35	ns
		4-mA drive		14	25	ns
		8-mA drive		13	24	ns
		8-mA drive with slew rate control		18	28	ns
J13 t_{TDO_DVZ}	TCK fall to High-Z from Data Valid	2-mA drive	-	9	11	ns
		4-mA drive		7	9	ns
		8-mA drive		6	8	ns
		8-mA drive with slew rate control		7	9	ns

Figure 21-2. JTAG Test Clock Input Timing

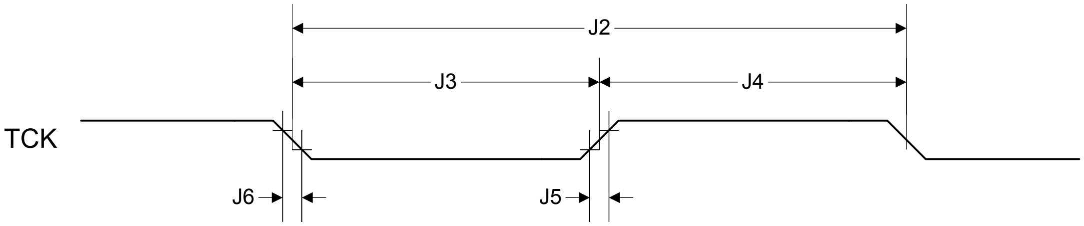
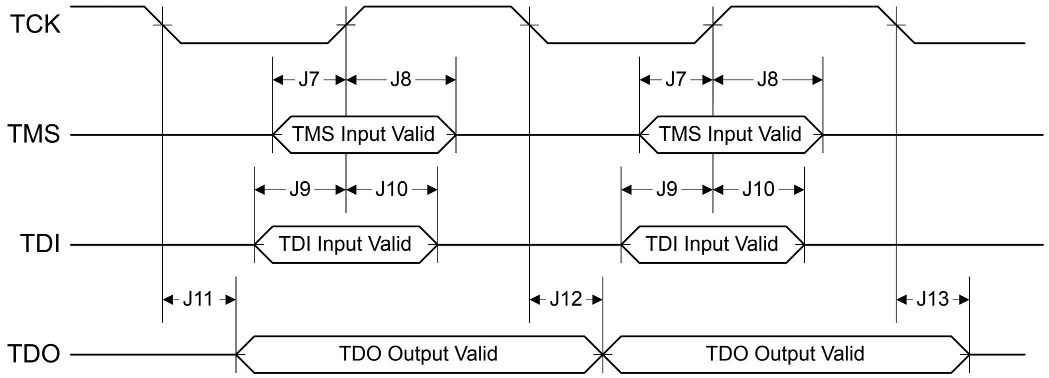


Figure 21-3. JTAG Test Access Port (TAP) Timing



21.2.4 Reset

Table 21-18. 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	-	-	95	system clocks
R6	T_{IRBOR}	Internal reset timeout after BOR	-	-	7	system clocks
R7	T_{IRHWR}	Internal reset timeout after hardware reset ($\overline{RST}$ pin)	-	-	7	system clocks
R8	T_{IRSWR}	Internal reset timeout after software-initiated system reset	-	-	16	system clocks
R9	T_{IRWDR}	Internal reset timeout after watchdog reset	-	-	16	system clocks
R10	T_{IRMFR}	Internal reset timeout after MOSC failure reset	-	-	32	system clocks
R11	$T_{VDDRISE}$	Supply voltage (V_{DD}) rise time (0V-3.3V)	-	-	250	ms
R12	T_{MIN}	Minimum $\overline{RST}$ pulse width	2	-	-	µs

Figure 21-4. External Reset Timing ($\overline{RST}$)

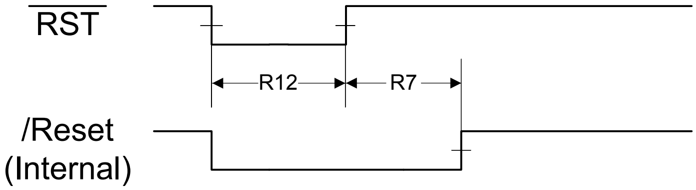


Figure 21-5. Power-On Reset Timing

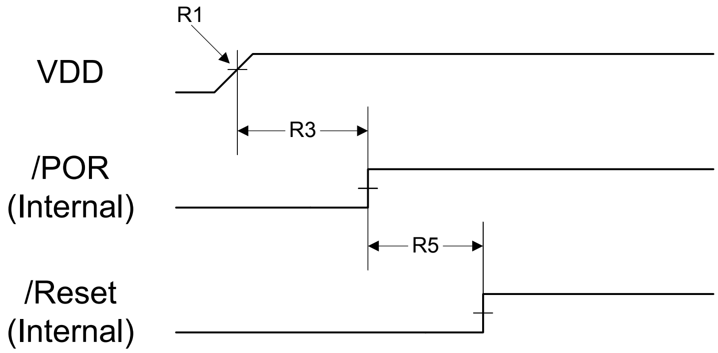


Figure 21-6. Brown-Out Reset Timing

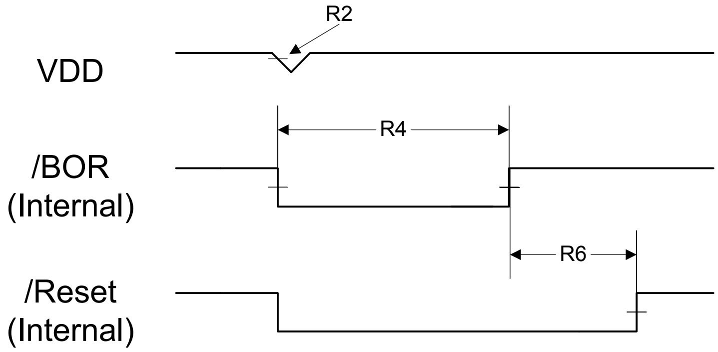


Figure 21-7. Software Reset Timing

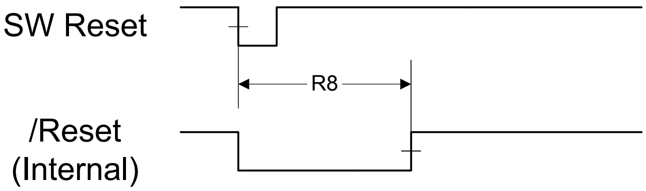


Figure 21-8. Watchdog Reset Timing

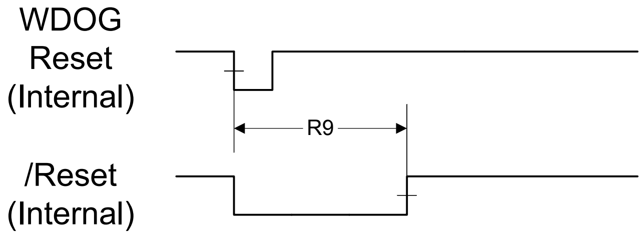
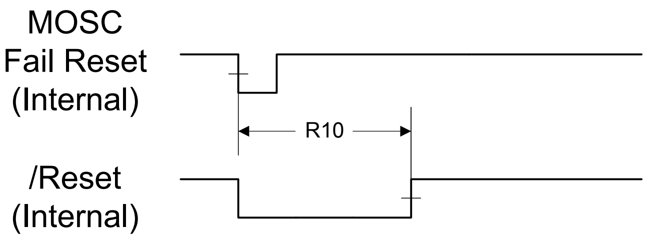


Figure 21-9. MOSC Failure Reset Timing



21.2.5 Deep-Sleep Mode

Table 21-19. Deep-Sleep Mode AC Characteristics

Parameter No	Parameter	Parameter Name	Min	Nom	Max	Unit
D1	t_{ENTER_DS}	Time to enter deep-sleep mode from sleep request	-	0	16[a]	ms

a. Nominal specification occurs 99.9995% of the time.

21.2.6 Hibernation Module

The Hibernation Module requires special system implementation considerations because it is intended to power down all other sections of its host device, refer to "Hibernation Module" on page 168.

Table 21-20. Hibernation Module AC Characteristics

Parameter No	Parameter	Parameter Name	Min	Nom	Max	Unit
H1	t_{HIB_LOW}	Internal 32.768 KHz clock reference rising edge to $\overline{\text{HIB}}$ asserted	20	-	-	µs
H2	t_{HIB_HIGH}	Internal 32.768 KHz clock reference rising edge to $\overline{\text{HIB}}$ deasserted	-	30	-	µs
H3	$t_{WAKE\_TO\_HIB}$	$\overline{\text{WAKE}}$ assert to $\overline{\text{HIB}}$ desassert (wake up time), internal Hibernation oscillator running during hibernation	62	-	124	µs
H4	$t_{WAKE\_TO\_HIB}$	$\overline{\text{WAKE}}$ assert to $\overline{\text{HIB}}$ desassert (wake up time), internal Hibernation oscillator stopped during hibernation	-	-	10	ms
H5	t_{WAKE_CLOCK}	$\overline{\text{WAKE}}$ assertion time, internal Hibernation oscillator running during hibernation	62	-	-	µs
H6	$t_{WAKE\_NOCLOCK}$	$\overline{\text{WAKE}}$ assertion time, internal Hibernation oscillator stopped during hibernation[a]	10	-	-	ms
H7	$t_{HIB\_REG\_ACCESS}$	Access time to or from a non-volatile register in HIB module to complete	92	-	-	µs
H8	$t_{HIB\_TO\_HIB}$	$\overline{\text{HIB}}$ high time between assertions	100	-	-	ms
H9	t_{ENTER_HIB}	Time to enter hibernation mode from hibernation request	-	0	50[b]	ms

a. This mode is used when the `PINWEN` bit is set and the `RTCEN` bit is clear in the **HIBCTL** register.

b. Nominal specification occurs 99.998% of the time.

Figure 21-10. Hibernation Module Timing with Internal Oscillator Running in Hibernation

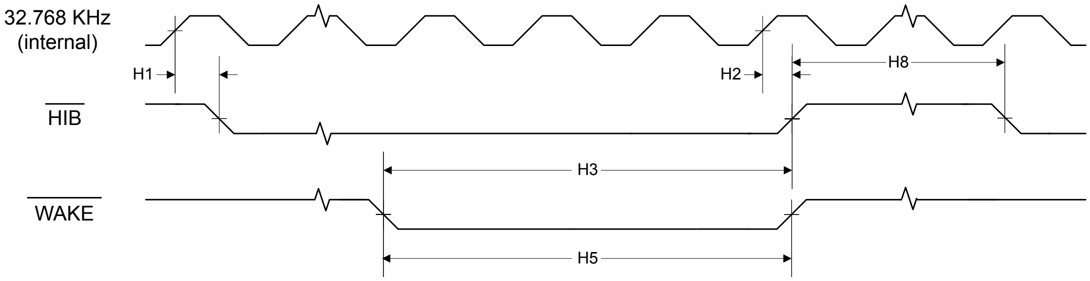
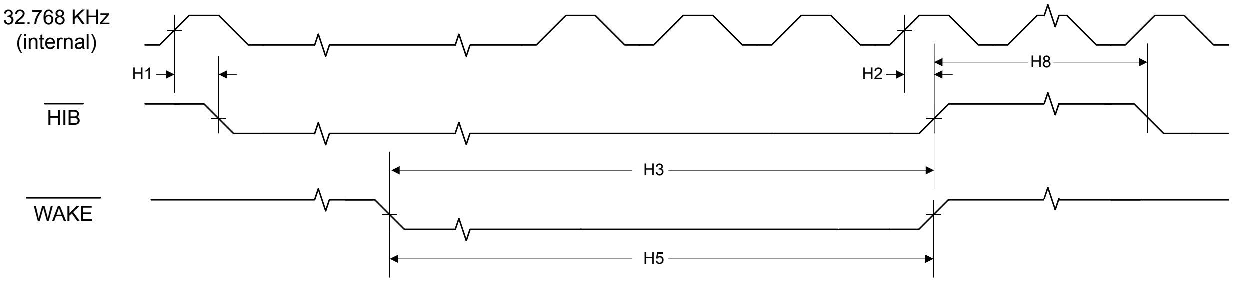


Figure 21-11. Hibernation Module Timing with Internal Oscillator Stopped in Hibernation



21.2.7 General-Purpose I/O (GPIO)

Note: All GPIOs are 5-V tolerant.

Table 21-21. GPIO Characteristics

Parameter	Parameter Name	Condition	Min	Nom	Max	Unit
t_{GPIOR}	GPIO Rise Time (from 20% to 80% of V_{DD})	2-mA drive	-	14	20	ns
		4-mA drive		7	10	ns
		8-mA drive		4	5	ns
		8-mA drive with slew rate control		6	8	ns
t_{GPIOF}	GPIO Fall Time (from 80% to 20% of V_{DD})	2-mA drive	-	14	21	ns
		4-mA drive		7	11	ns
		8-mA drive		4	6	ns
		8-mA drive with slew rate control		6	8	ns

21.2.8 Analog-to-Digital Converter

Table 21-22. ADC Characteristics[a]

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{ADCIN}	Maximum single-ended, full-scale analog input voltage	-	-	3.0	V
	Minimum single-ended, full-scale analog input voltage	0.0	-	-	V
	Maximum differential, full-scale analog input voltage	-	-	1.5	V
	Minimum differential, full-scale analog input voltage	0.0	-	-	V
N	Resolution	10			bits

Table 21-22. ADC Characteristics *(continued)*

Parameter	Parameter Name	Min	Nom	Max	Unit
f_{ADC}	ADC internal clock frequency[b]	14	16	18	MHz
$t_{ADCCONV}$	Conversion time[c]	1			µs
$f_{ADCCONV}$	Conversion rate[c]	1000			k samples/s
t_{LT}	Latency from trigger to start of conversion	-	2	-	system clocks
I_L	ADC input leakage	-	-	±1.0	µA
R_{ADC}	ADC equivalent resistance	-	-	10	kΩ
C_{ADC}	ADC equivalent capacitance	0.9	1.0	1.1	pF
E_L	Integral nonlinearity error	-	-	±1	LSB
E_D	Differential nonlinearity error	-	-	±1	LSB
E_O	Offset error	-	-	±1	LSB
E_G	Full-scale gain error	-	-	±3	LSB
E_{TS}	Temperature sensor accuracy	-	-	±5	°C

[a] The ADC reference voltage is 3.0 V. This reference voltage is internally generated from the 3.3 VDDA supply by a band gap circuit.

[b] The ADC must be clocked from the PLL or directly from an external clock source to operate properly.

[c] The conversion time and rate scale from the specified number if the ADC internal clock frequency is any value other than 16 MHz.

Figure 21-12. ADC Input Equivalency Diagram

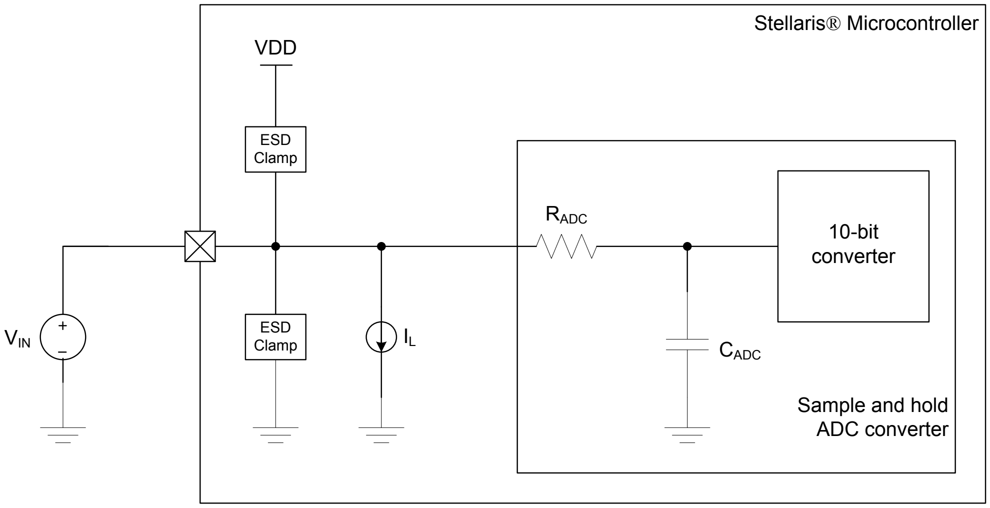


Table 21-23. ADC Module External Reference Characteristics[a]

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{REFA}	External voltage reference for ADC[b]	2.4	-	V_{DD}	V
I_L	External voltage reference leakage current	-	±1.0	-	µA

[a] Care must be taken to supply a reference voltage of acceptable quality.

[b] Ground is always used as the reference level for the minimum conversion value.

Table 21-24. ADC Module Internal Reference Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{REFI}	Internal voltage reference for ADC	-	3.0	-	V
E_{IR}	Internal voltage reference error	-	-	±2.5	%

21.2.9 Synchronous Serial Interface (SSI)

Table 21-25. SSI Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
S1	t_{CLK_PER}	`SSIClk` cycle time	2	-	65024	system clocks
S2	t_{CLK_HIGH}	`SSIClk` high time	-	0.5	-	t clk_per
S3	t_{CLK_LOW}	`SSIClk` low time	-	0.5	-	t clk_per
S4	t_{CLKRF}	`SSIClk` rise/fall time	-	7.4	26	ns
S5	t_{DMD}	Data from master valid delay time	0	-	1	system clocks
S6	t_{DMS}	Data from master setup time	1	-	-	system clocks
S7	t_{DMH}	Data from master hold time	2	-	-	system clocks
S8	t_{DSS}	Data from slave setup time	1	-	-	system clocks
S9	t_{DSH}	Data from slave hold time	2	-	-	system clocks

Figure 21-13. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement

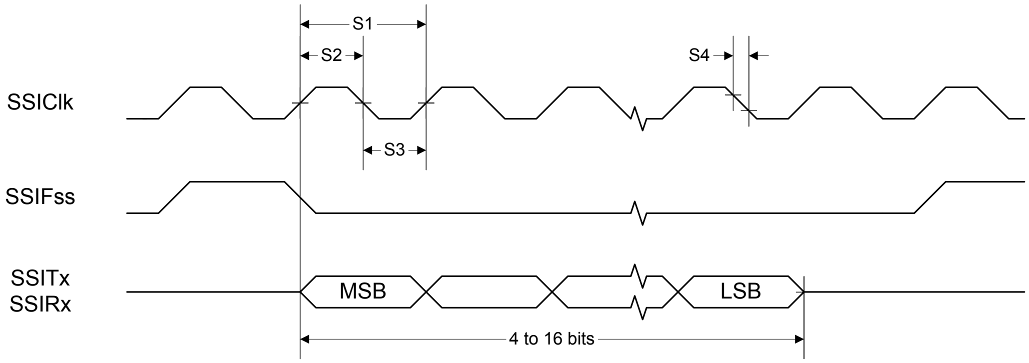


Figure 21-14. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer

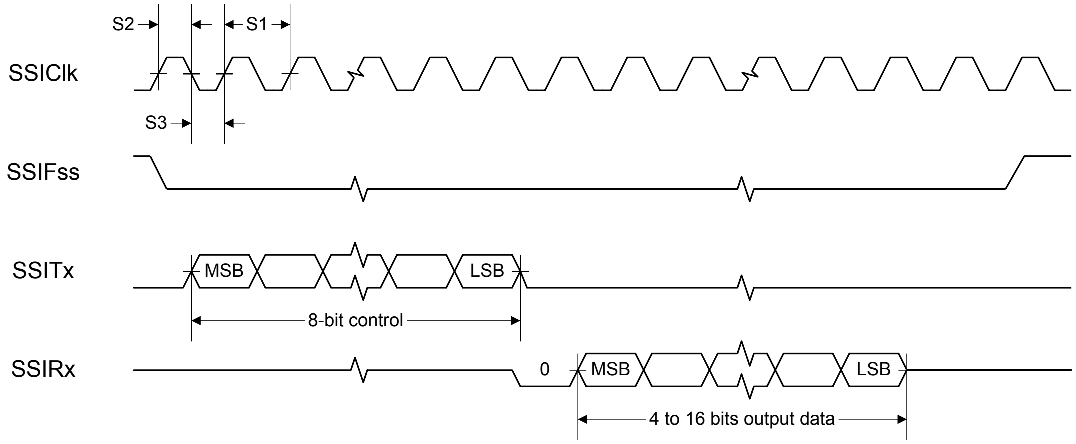
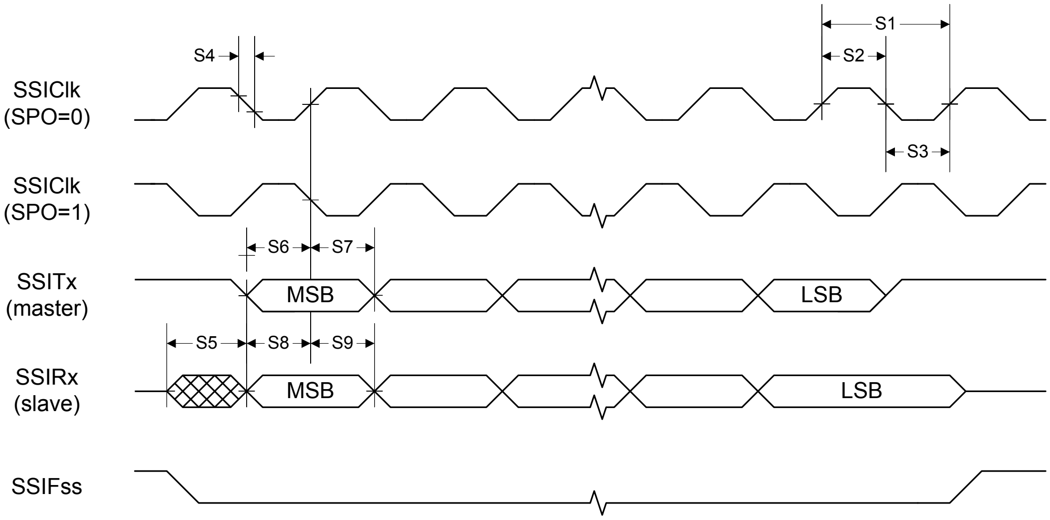
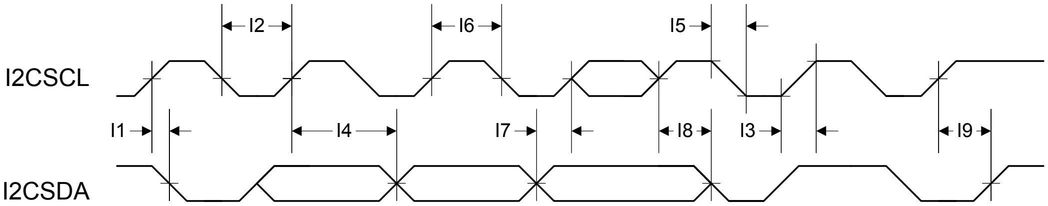


Figure 21-15. SSI Timing for SPI Frame Format (FRF=00), with SPH=1



21.2.10 Inter-Integrated Circuit (I^2C) Interface

Figure 21-16. I^2C Timing



21.2.11 Analog Comparator

Table 21-26. 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 21-27. Analog Comparator Voltage Reference Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
R_{HR}	Resolution high range	-	V_{DD}/31	-	LSB
R_{LR}	Resolution low range	-	V_{DD}/23	-	LSB
A_{HR}	Absolute accuracy high range	-	-	±1/2	LSB
A_{LR}	Absolute accuracy low range	-	-	±1/4	LSB

A Boot Loader

A.1 Boot Loader Overview

The Stellaris® Boot Loader is executed from the ROM when the Flash memory is empty and is used to download code to the Flash memory of a device without the use of a debug interface. The boot loader uses a simple packet interface to provide synchronous communication with the device. The speed of the boot loader is determined by the internal oscillator (PIOSC) frequency as it does not enable the PLL. The following serial interfaces can be used:

- UART0
- SSI0
- I²C0

For simplicity, both the data format and communication protocol are identical for all serial interfaces.

See the *Stellaris® Boot Loader User's Guide* for information on the boot loader software.

A.2 Serial Interfaces

This section describes how the boot loader operates using a serial interface.

A.2.1 Serial Configuration

Once communication with the boot loader is established via one of the serial interfaces, that interface is used until the boot loader is reset or new code takes over. For example, once you start communicating using the SSI port, communications with the boot loader via the UART are disabled until the device is reset.

A.2.1.1 UART

The Universal Asynchronous Receivers/Transmitters (UART) communication uses a fixed serial format of 8 bits of data, no parity, and 1 stop bit. The baud rate used for communication is automatically detected by the boot loader and can be any valid baud rate supported by the host and the device. The auto detection sequence requires that the baud rate should be no more than 1/32 the internal oscillator (PIOSC) frequency of the board that is running the boot loader (which is at least 8.4 MHz, providing support for up to 262,500 baud). The maximum regular speed baud rate for any UART on a Stellaris® device is calculated as follows:

```
Max Baud Rate = System Clock Frequency / 16
```

In order to determine the baud rate, the boot loader must determine the relationship between the internal oscillator and the baud rate. With this information, the boot loader can configure the UART to the same baud rate as the host. This automatic baud-rate detection allows the host to use any valid baud rate to communicate with the device.

The method used to perform this automatic synchronization requires the host to send the boot loader two bytes that are both 0x55. With this series of pulses, the boot loader can calculate the ratios needed to program the UART to match the host's baud rate. After the host sends the pattern, it attempts to read back one byte of data from the UART. The boot loader returns the value of 0xCC to indicate successful detection of the baud rate. If this byte is not received after at least twice the time required to transfer the two bytes, the host can resend another pattern of 0x55, 0x55, and wait for the 0xCC byte again until the boot loader acknowledges that it has received a synchronization pattern correctly. For example, the time to wait for data back from the boot loader should be calculated as at least 2*(20(bits/sync)/baud rate (bits/sec)). For a baud rate of 115200, this time is 2*(20/115200) or 0.35 ms.

A.2.1.2 SSI

The Synchronous Serial Interface (SSI) port also uses a fixed serial format for communications, with the default framing defined as Motorola format with both the `SPH` and `SPO` bits set in the **SSICR0** register. See “Frame Formats” on page 539 for more information on formats for this transfer protocol. Like the UART, this interface has hardware requirements that limit the maximum frequency of the SSIClk signal to be at most 1/12 the internal oscillator (PIOSC) frequency of the board running the boot loader (which is at least 8.4 MHz, providing support for up to 700 KHz). Because the host device is the master, the SSI on the boot loader device does not need to determine the clock as it is provided directly by the host.

A.2.1.3 I^{2}C

The Inter-Integrated Circuit (I^{2}C) port operates in slave mode with a slave address of 0x42. The I^{2}C port works at both 100-kHz and 400-kHz `I2CSCL` clock frequency. Because the host device is the master, the I^{2}C on the boot loader device does not need to determine the clock as it is provided directly by the host.

A.2.2 Serial 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.2.2.1 Packet Format

All packets sent and received from the device use the following byte-packed format.

```
struct
{
 unsigned char ucSize;
 unsigned char ucCheckSum;
 unsigned char Data[];
};
```

ucSize	The first byte received holds the total size of the transfer including the size and checksum bytes.
ucChecksum	This holds a simple checksum of the bytes in the data buffer only. The algorithm is Data[0]+Data[1]+…+ Data[`ucSize`-3].
Data	This is the raw data intended for the device, which is formatted in some form of command interface. There should be `ucSize`–2 bytes of data provided in this buffer to or from the device.

A.2.2.2 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 memory programming. This limitation is discussed further in the section that describes the boot loader command, `COMMAND_SEND_DATA` (see “COMMAND_SEND_DATA (0x24)” on page 661).

Once the packet has been formatted correctly by the host, it should be sent out over the serial interface. Then the host should poll the interface for the first non-zero data returned from the device. The first non-zero byte is either an ACK (0xCC) or a NAK (0x33) byte from the device indicating the packet was received successfully (ACK) or unsuccessfully (NAK). This response 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.2.2.3 Receiving Packets

The boot loader sends a packet of data in the same format that it receives a packet. The boot loader may transfer leading zero data before the first actual byte of data is sent out. The first non-zero byte is the size of the packet followed by a checksum byte and finally followed by the data itself. The data is sent without a break after the first non-zero byte is sent from the boot loader. Once the device communicating with the boot loader receives all the bytes, it must either ACK or NAK the packet to indicate that the transmission was successful. The appropriate response after sending a NAK to the boot loader is to resend the command that failed and request the data again. If needed, the host may send leading zeros before sending down the ACK/NAK signal to the boot loader, as the boot loader only accepts the first non-zero data as a valid response. This zero padding is needed by the SSI interface in order to receive data to or from the boot loader.

A.2.3 Serial Commands

The next section defines the list of commands that can be sent to the boot loader. The first byte of the data should always be one of the defined commands, followed by data or parameters as determined by the command that is sent.

A.2.3.1 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. Because the ping command has no real return status, the receipt of an ACK can be interpreted as a successful ping to the boot loader.

A.2.3.2 COMMAND_DOWNLOAD (0x21)

This command is sent to the boot loader to indicate where to store data and how many bytes will be sent by the `COMMAND_SEND_DATA` commands that follow. The command consists of two 32-bit values that are both transferred MSB first. The first 32-bit value is the address to start programming data into, while the second is the 32-bit size of the data that will be sent. This command also triggers an erase of the full area to be programmed so this command takes longer than other commands and results in a longer time to receive the ACK/NAK back from the board. This command should be followed by a `COMMAND_GET_STATUS` to ensure that the Program Address and Program size are valid for the device running the boot loader.

The format of the packet to send this command is 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.2.3.3 COMMAND_RUN (0x22)

This command is used to tell the boot loader to execute from the address passed as the parameter in this command. This command consists of a single 32-bit value that is interpreted as the address to execute. The 32-bit value is transmitted MSB first and the boot loader responds with an ACK signal back to the host device before actually executing the code at the given address. The ACK response tells the host that the command was received successfully, and the code is 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.2.3.4 COMMAND_GET_STATUS (0x23)

This command returns the status of the last command that was issued. Typically, this command should be sent after every command to ensure that the previous command was successful or to properly respond to a failure. The command requires one byte in the data of the packet and should be followed by reading a packet with one byte of data that contains a status code. The last step is to ACK or NAK the received data so the boot loader knows that the data has been read.

```
Byte[0] = 0x03
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_GET_STATUS
```

A.2.3.5 COMMAND_SEND_DATA (0x24)

This command should only follow a `COMMAND_DOWNLOAD` command or another `COMMAND_SEND_DATA` command if more data is needed. Consecutive send data commands automatically increment address and continue programming from the previous location. For packets which do not contain the final portion of the downloaded data, a multiple of four bytes should always be transferred. The command terminates programming once the number of bytes indicated by the `COMMAND_DOWNLOAD` command has been received. Each time this function is called, it should be followed by a `COMMAND_GET_STATUS` to ensure that the data was successfully programmed into the Flash memory. If the boot loader sends a NAK to this command, the boot loader does not increment the current address to allow retransmission of the previous data. The following example shows a `COMMAND_SEND_DATA` packet with 8 bytes of packet data:

```
Byte[0] = 11
Byte[1] = checksum(Bytes[2:10])
Byte[2] = COMMAND_SEND_DATA
Byte[3] = Data[0]
Byte[4] = Data[1]
Byte[5] = Data[2]
Byte[6] = Data[3]
Byte[7] = Data[4]
Byte[8] = Data[5]
Byte[9] = Data[6]
Byte[10] = Data[7]
```

A.2.3.6 COMMAND_RESET (0x25)

This command is used to tell the boot loader device to reset. Unlike the `COMMAND_RUN`, this command allows the initial stack pointer to be read by the hardware and set up for the new code. `COMMAND_RESET` can also be used to reset the boot loader if a critical error occurs, and the host device wants to restart communication with the boot loader.

```
Byte[0] = 3
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_RESET
```

The boot loader responds with an ACK signal back to the host device before actually executing the software reset to the device running the boot loader. The ACK tells the host that the command was received successfully and the part will be reset.

B ROM DriverLib Functions

B.1 DriverLib Functions Included in the Integrated ROM

The Stellaris® Peripheral Driver Library (DriverLib) APIs that are available in the integrated ROM of the Stellaris® family of devices are listed below. The detailed description of each function is available in the *Stellaris® ROM User's Guide*.

ROM_ADCHardwareOversampleConfigure
// Configures the hardware oversampling factor of the ADC.

ROM_ADCIntClear
// Clears sample sequence interrupt source.

ROM_ADCIntDisable
// Disables a sample sequence interrupt.

ROM_ADCIntEnable
// Enables a sample sequence interrupt.

ROM_ADCIntStatus
// Gets the current interrupt status.

ROM_ADCProcessorTrigger
// Causes a processor trigger for a sample sequence.

ROM_ADCSequenceConfigure
// Configures the trigger source and priority of a sample sequence.

ROM_ADCSequenceDataGet
// Gets the captured data for a sample sequence.

ROM_ADCSequenceDisable
// Disables a sample sequence.

ROM_ADCSequenceEnable
// Enables a sample sequence.

ROM_ADCSequenceOverflow
// Determines if a sample sequence overflow occurred.

ROM_ADCSequenceOverflowClear
// Clears the overflow condition on a sample sequence.

ROM_ADCSequenceStepConfigure
// Configure a step of the sample sequencer.

ROM_ADCSequenceUnderflow
// Determines if a sample sequence underflow occurred.

ROM_ADCSequenceUnderflowClear
// Clears the underflow condition on a sample sequence.

```
ROM_ComparatorConfigure
    // Configures a comparator.

ROM_ComparatorIntClear
    // Clears a comparator interrupt.

ROM_ComparatorIntDisable
    // Disables the comparator interrupt.

ROM_ComparatorIntEnable
    // Enables the comparator interrupt.

ROM_ComparatorIntStatus
    // Gets the current interrupt status.

ROM_ComparatorRefSet
    // Sets the internal reference voltage.

ROM_ComparatorValueGet
    // Gets the current comparator output value.

ROM_Crc16Array
    // Calculates the CRC-16 of an array of words.

ROM_Crc16Array3
    // Calculates three CRC-16s of an array of words.

ROM_FlashErase
    // Erases a block of flash.

ROM_FlashIntClear
    // Clears flash controller interrupt sources.

ROM_FlashIntDisable
    // Disables individual flash controller interrupt sources.

ROM_FlashIntEnable
    // Enables individual flash controller interrupt sources.

ROM_FlashIntGetStatus
    // Gets the current interrupt status.

ROM_FlashProgram
    // Programs flash.

ROM_FlashProtectGet
    // Gets the protection setting for a block of flash.

ROM_FlashProtectSave
    // Saves the flash protection settings.

ROM_FlashProtectSet
    // Sets the protection setting for a block of flash.
```

ROM_FlashUsecGet
// Gets the number of processor clocks per micro-second.

ROM_FlashUsecSet
// Sets the number of processor clocks per micro-second.

ROM_FlashUserGet
// Gets the user registers.

ROM_FlashUserSave
// Saves the user registers.

ROM_FlashUserSet
// Sets the user registers.

ROM_GPIODirModeGet
// Gets the direction and mode of a pin.

ROM_GPIODirModeSet
// Sets the direction and mode of the specified pin(s).

ROM_GPIOIntTypeGet
// Gets the interrupt type for a pin.

ROM_GPIOIntTypeSet
// Sets the interrupt type for the specified pin(s).

ROM_GPIOPadConfigGet
// Gets the pad configuration for a pin.

ROM_GPIOPadConfigSet
// Sets the pad configuration for the specified pin(s).

ROM_GPIOPinConfigure
// Configures the alternate function of a GPIO pin.

ROM_GPIOPinIntClear
// Clears the interrupt for the specified pin(s).

ROM_GPIOPinIntDisable
// Disables interrupts for the specified pin(s).

ROM_GPIOPinIntEnable
// Enables interrupts for the specified pin(s).

ROM_GPIOPinIntStatus
// Gets interrupt status for the specified GPIO port.

ROM_GPIOPinRead
// Reads the values present of the specified pin(s).

ROM_GPIOPinTypeADC
// Configures pin(s) for use as analog-to-digital converter inputs.

```
ROM_GPIOPinTypeComparator
    // Configures pin(s) for use as an analog comparator input.

ROM_GPIOPinTypeGPIOInput
    // Configures pin(s) for use as GPIO inputs.

ROM_GPIOPinTypeGPIOOutput
    // Configures pin(s) for use as GPIO outputs.

ROM_GPIOPinTypeGPIOOutputOD
    // Configures pin(s) for use as GPIO open drain outputs.

ROM_GPIOPinTypeI2C
    // Configures pin(s) for use by the I2C peripheral.

ROM_GPIOPinTypeSSI
    // Configures pin(s) for use by the SSI peripheral.

ROM_GPIOPinTypeTimer
    // Configures pin(s) for use by the Timer peripheral.

ROM_GPIOPinTypeUART
    // Configures pin(s) for use by the UART peripheral.

ROM_GPIOPinWrite
    // Writes a value to the specified pin(s).

ROM_HibernateClockSelect
    // Selects the clock input for the Hibernation module.

ROM_HibernateDataGet
    // Reads a set of data from the non-volatile memory of the Hibernation module.

ROM_HibernateDataSet
    // Stores data in the non-volatile memory of the Hibernation module.

ROM_HibernateDisable
    // Disables the Hibernation module for operation.

ROM_HibernateEnableExpClk
    // Enables the Hibernation module for operation.

ROM_HibernateIntClear
    // Clears pending interrupts from the Hibernation module.

ROM_HibernateIntDisable
    // Disables interrupts for the Hibernation module.

ROM_HibernateIntEnable
    // Enables interrupts for the Hibernation module.

ROM_HibernateIntStatus
    // Gets the current interrupt status of the Hibernation module.
```

ROM_HibernateIsActive
// Checks to see if the Hibernation module is already powered up.

ROM_HibernateLowBatGet
// Gets the currently configured low battery detection behavior.

ROM_HibernateLowBatSet
// Configures the low battery detection.

ROM_HibernateRequest
// Requests hibernation mode.

ROM_HibernateRTCDisable
// Disables the RTC feature of the Hibernation module.

ROM_HibernateRTCEnable
// Enables the RTC feature of the Hibernation module.

ROM_HibernateRTCGet
// Gets the value of the real time clock (RTC) counter.

ROM_HibernateRTCMatch0Get
// Gets the value of the RTC match 0 register.

ROM_HibernateRTCMatch0Set
// Sets the value of the RTC match 0 register.

ROM_HibernateRTCMatch1Get
// Gets the value of the RTC match 1 register.

ROM_HibernateRTCMatch1Set
// Sets the value of the RTC match 1 register.

ROM_HibernateRTCSet
// Sets the value of the real time clock (RTC) counter.

ROM_HibernateRTCTrimGet
// Gets the value of the RTC predivider trim register.

ROM_HibernateRTCTrimSet
// Sets the value of the RTC predivider trim register.

ROM_HibernateWakeGet
// Gets the currently configured wake conditions for the Hibernation module.

ROM_HibernateWakeSet
// Configures the wake conditions for the Hibernation module.

ROM_I2CMasterBusBusy
// Indicates whether or not the I2C bus is busy.

ROM_I2CMasterBusy
// Indicates whether or not the I2C Master is busy.

```
ROM_I2CMasterControl
    // Controls the state of the I2C Master module.

ROM_I2CMasterDataGet
    // Receives a byte that has been sent to the I2C Master.

ROM_I2CMasterDataPut
    // Transmits a byte from the I2C Master.

ROM_I2CMasterDisable
    // Disables the I2C master block.

ROM_I2CMasterEnable
    // Enables the I2C Master block.

ROM_I2CMasterErr
    // Gets the error status of the I2C Master module.

ROM_I2CMasterInitExpClk
    // Initializes the I2C Master block.

ROM_I2CMasterIntClear
    // Clears I2C Master interrupt sources.

ROM_I2CMasterIntDisable
    // Disables the I2C Master interrupt.

ROM_I2CMasterIntEnable
    // Enables the I2C Master interrupt.

ROM_I2CMasterIntStatus
    // Gets the current I2C Master interrupt status.

ROM_I2CMasterSlaveAddrSet
    // Sets the address that the I2C Master will place on the bus.

ROM_I2CSlaveDataGet
    // Receives a byte that has been sent to the I2C Slave.

ROM_I2CSlaveDataPut
    // Transmits a byte from the I2C Slave.

ROM_I2CSlaveDisable
    // Disables the I2C slave block.

ROM_I2CSlaveEnable
    // Enables the I2C Slave block.

ROM_I2CSlaveInit
    // Initializes the I2C Slave block.

ROM_I2CSlaveIntClear
    // Clears I2C Slave interrupt sources.
```

```
ROM_I2CSlaveIntClearEx
    // Clears I2C Slave interrupt sources.

ROM_I2CSlaveIntDisable
    // Disables the I2C Slave interrupt.

ROM_I2CSlaveIntDisableEx
    // Disables individual I2C Slave interrupt sources.

ROM_I2CSlaveIntEnable
    // Enables the I2C Slave interrupt.

ROM_I2CSlaveIntEnableEx
    // Enables individual I2C Slave interrupt sources.

ROM_I2CSlaveIntStatus
    // Gets the current I2C Slave interrupt status.

ROM_I2CSlaveIntStatusEx
    // Gets the current I2C Slave interrupt status.

ROM_I2CSlaveStatus
    // Gets the I2C Slave module status.

ROM_IntDisable
    // Disables an interrupt.

ROM_IntEnable
    // Enables an interrupt.

ROM_IntMasterDisable
    // Disables the processor interrupt.

ROM_IntMasterEnable
    // Enables the processor interrupt.

ROM_IntPriorityGet
    // Gets the priority of an interrupt.

ROM_IntPriorityGroupingGet
    // Gets the priority grouping of the interrupt controller.

ROM_IntPriorityGroupingSet
    // Sets the priority grouping of the interrupt controller.

ROM_IntPrioritySet
    // Sets the priority of an interrupt.

ROM_MPUDisable
    // Disables the MPU for use.

ROM_MPUEnable
    // Enables and configures the MPU for use.
```

ROM_MPURegionCountGet
    // Gets the count of regions supported by th MPU.

ROM_MPURegionDisable
    // Disables a specific region.

ROM_MPURegionEnable
    // Enables a specific region.

ROM_MPURegionGet
    // Gets the current settings for a specific region.

ROM_MPURegionSet
    // Sets up the access rules for a specific region.

ROM_SSIConfigSetExpClk
    // Configures the synchronous serial interface.

ROM_SSIDataGet
    // Gets a data element from the SSI receive FIFO.

ROM_SSIDataGetNonBlocking
    // Gets a data element from the SSI receive FIFO.

ROM_SSIDataPut
    // Puts a data element into the SSI transmit FIFO.

ROM_SSIDataPutNonBlocking
    // Puts a data element into the SSI transmit FIFO.

ROM_SSIDisable
    // Disables the synchronous serial interface.

ROM_SSIDMADisable
    // Disable SSI DMA operation.

ROM_SSIDMAEnable
    // Enable SSI DMA operation.

ROM_SSIEnable
    // Enables the synchronous serial interface.

ROM_SSIIntClear
    // Clears SSI interrupt sources.

ROM_SSIIntDisable
    // Disables individual SSI interrupt sources.

ROM_SSIIntEnable
    // Enables individual SSI interrupt sources.

ROM_SSIIntStatus
    // Gets the current interrupt status.

```
ROM_SysCtlADCSpeedGet
    // Gets the sample rate of the ADC.

ROM_SysCtlADCSpeedSet
    // Sets the sample rate of the ADC.

ROM_SysCtlClockGet
    // Gets the processor clock rate.

ROM_SysCtlClockSet
    // Sets the clocking of the device.

ROM_SysCtlDeepSleep
    // Puts the processor into deep-sleep mode.

ROM_SysCtlDelay
    // Provides a small delay.

ROM_SysCtlFlashSizeGet
    // Gets the size of the flash.

ROM_SysCtlGPIOAHBDisable
    // Disables a GPIO peripheral for access from the AHB.

ROM_SysCtlGPIOAHBEnable
    // Enables a GPIO peripheral for access from the AHB.

ROM_SysCtlIntClear
    // Clears system control interrupt sources.

ROM_SysCtlIntDisable
    // Disables individual system control interrupt sources.

ROM_SysCtlIntEnable
    // Enables individual system control interrupt sources.

ROM_SysCtlIntStatus
    // Gets the current interrupt status.

ROM_SysCtlLDOGet
    // Gets the output voltage of the LDO.

ROM_SysCtlLDOSet
    // Sets the output voltage of the LDO.

ROM_SysCtlPeripheralClockGating
    // Controls peripheral clock gating in sleep and deep-sleep mode.

ROM_SysCtlPeripheralDeepSleepDisable
    // Disables a peripheral in deep-sleep mode.

ROM_SysCtlPeripheralDeepSleepEnable
    // Enables a peripheral in deep-sleep mode.
```

ROM_SysCtlPeripheralDisable
// Disables a peripheral.

ROM_SysCtlPeripheralEnable
// Enables a peripheral.

ROM_SysCtlPeripheralPresent
// Determines if a peripheral is present.

ROM_SysCtlPeripheralReset
// Performs a software reset of a peripheral.

ROM_SysCtlPeripheralSleepDisable
// Disables a peripheral in sleep mode.

ROM_SysCtlPeripheralSleepEnable
// Enables a peripheral in sleep mode.

ROM_SysCtlPinPresent
// Determines if a pin is present.

ROM_SysCtlReset
// Resets the device.

ROM_SysCtlResetCauseClear
// Clears reset reasons.

ROM_SysCtlResetCauseGet
// Gets the reason for a reset.

ROM_SysCtlSleep
// Puts the processor into sleep mode.

ROM_SysCtlSRAMSizeGet
// Gets the size of the SRAM.

ROM_SysTickDisable
// Disables the SysTick counter.

ROM_SysTickEnable
// Enables the SysTick counter.

ROM_SysTickIntDisable
// Disables the SysTick interrupt.

ROM_SysTickIntEnable
// Enables the SysTick interrupt.

ROM_SysTickPeriodGet
// Gets the period of the SysTick counter.

ROM_SysTickPeriodSet
// Sets the period of the SysTick counter.

ROM_SysTickValueGet
// Gets the current value of the SysTick counter.

ROM_TimerConfigure
// Configures the timer(s).

ROM_TimerControlEvent
// Controls the event type.

ROM_TimerControlLevel
// Controls the output level.

ROM_TimerControlStall
// Controls the stall handling.

ROM_TimerControlTrigger
// Enables or disables the trigger output.

ROM_TimerDisable
// Disables the timer(s).

ROM_TimerEnable
// Enables the timer(s).

ROM_TimerIntClear
// Clears timer interrupt sources.

ROM_TimerIntDisable
// Disables individual timer interrupt sources.

ROM_TimerIntEnable
// Enables individual timer interrupt sources.

ROM_TimerIntStatus
// Gets the current interrupt status.

ROM_TimerLoadGet
// Gets the timer load value.

ROM_TimerLoadSet
// Sets the timer load value.

ROM_TimerMatchGet
// Gets the timer match value.

ROM_TimerMatchSet
// Sets the timer match value.

ROM_TimerPrescaleGet
// Get the timer prescale value.

ROM_TimerPrescaleSet
// Set the timer prescale value.

```
ROM_TimerRTCDisable
    // Disable RTC counting.

ROM_TimerRTCEnable
    // Enable RTC counting.

ROM_TimerValueGet
    // Gets the current timer value.

ROM_UARTBreakCtl
    // Causes a BREAK to be sent.

ROM_UARTBusy
    // Determines whether the UART transmitter is busy or not.

ROM_UARTCharGet
    // Waits for a character from the specified port.

ROM_UARTCharGetNonBlocking
    // Receives a character from the specified port.

ROM_UARTCharPut
    // Waits to send a character from the specified port.

ROM_UARTCharPutNonBlocking
    // Sends a character to the specified port.

ROM_UARTCharsAvail
    // Determines if there are any characters in the receive FIFO.

ROM_UARTConfigGetExpClk
    // Gets the current configuration of a UART.

ROM_UARTConfigSetExpClk
    // Sets the configuration of a UART.

ROM_UARTDisable
    // Disables transmitting and receiving.

ROM_UARTDisableSIR
    // Disables SIR (IrDA) mode on the specified UART.

ROM_UARTDMADisable
    // Disable UART DMA operation.

ROM_UARTDMAEnable
    // Enable UART DMA operation.

ROM_UARTEnable
    // Enables transmitting and receiving.

ROM_UARTEnableSIR
    // Enables SIR (IrDA) mode on the specified UART.
```

ROM_UARTFIFODisable
// Disables the transmit and receive FIFOs.

ROM_UARTFIFOEnable
// Enables the transmit and receive FIFOs.

ROM_UARTFIFOLevelGet
// Gets the FIFO level at which interrupts are generated.

ROM_UARTFIFOLevelSet
// Sets the FIFO level at which interrupts are generated.

ROM_UARTIntClear
// Clears UART interrupt sources.

ROM_UARTIntDisable
// Disables individual UART interrupt sources.

ROM_UARTIntEnable
// Enables individual UART interrupt sources.

ROM_UARTIntStatus
// Gets the current interrupt status.

ROM_UARTParityModeGet
// Gets the type of parity currently being used.

ROM_UARTParityModeSet
// Sets the type of parity.

ROM_UARTRxErrorClear
// Clears all reported receiver errors.

ROM_UARTRxErrorGet
// Gets current receiver errors.

ROM_UARTSpaceAvail
// Determines if there is any space in the transmit FIFO.

ROM_UARTTxIntModeGet
// Returns the current operating mode for the UART transmit interrupt.

ROM_UARTTxIntModeSet
// Sets the operating mode for the UART transmit interrupt.

ROM_uDMAChannelAttributeDisable
// Disables attributes of a uDMA channel.

ROM_uDMAChannelAttributeEnable
// Enables attributes of a uDMA channel.

ROM_uDMAChannelAttributeGet
// Gets the enabled attributes of a uDMA channel.

ROM_uDMAChannelControlSet
// Sets the control parameters for a uDMA channel.

ROM_uDMAChannelDisable
// Disables a uDMA channel for operation.

ROM_uDMAChannelEnable
// Enables a uDMA channel for operation.

ROM_uDMAChannelIsEnabled
// Checks if a uDMA channel is enabled for operation.

ROM_uDMAChannelModeGet
// Gets the transfer mode for a uDMA channel.

ROM_uDMAChannelRequest
// Requests a uDMA channel to start a transfer.

ROM_uDMAChannelSelectDefault
// Select the default peripheral for a set of uDMA channels.

ROM_uDMAChannelSelectSecondary
// Select the secondary peripheral for a set of uDMA channels.

ROM_uDMAChannelSizeGet
// Gets the current transfer size for a uDMA channel.

ROM_uDMAChannelTransferSet
// Sets the transfer parameters for a uDMA channel.

ROM_uDMAControlBaseGet
// Gets the base address for the channel control table.

ROM_uDMAControlBaseSet
// Sets the base address for the channel control table.

ROM_uDMADisable
// Disables the uDMA controller for use.

ROM_uDMAEnable
// Enables the uDMA controller for use.

ROM_uDMAErrorStatusClear
// Clears the uDMA error interrupt.

ROM_uDMAErrorStatusGet
// Gets the uDMA error status.

ROM_uDMAIntClear
// Clears uDMA interrupt status.

ROM_uDMAIntStatus
// Gets the uDMA controller channel interrupt status.

```
ROM_UpdateI2C
    // Starts an update over the I2C0 interface.

ROM_UpdateSSI
    // Starts an update over the SSI0 interface.

ROM_UpdateUART
    // Starts an update over the UART0 interface.

ROM_WatchdogEnable
    // Enables the watchdog timer.

ROM_WatchdogIntClear
    // Clears the watchdog timer interrupt.

ROM_WatchdogIntEnable
    // Enables the watchdog timer interrupt.

ROM_WatchdogIntStatus
    // Gets the current watchdog timer interrupt status.

ROM_WatchdogLock
    // Enables the watchdog timer lock mechanism.

ROM_WatchdogLockState
    // Gets the state of the watchdog timer lock mechanism.

ROM_WatchdogReloadGet
    // Gets the watchdog timer reload value.

ROM_WatchdogReloadSet
    // Sets the watchdog timer reload value.

ROM_WatchdogResetDisable
    // Disables the watchdog timer reset.

ROM_WatchdogResetEnable
    // Enables the watchdog timer reset.

ROM_WatchdogRunning
    // Determines if the watchdog timer is enabled.

ROM_WatchdogStallDisable
    // Disables stalling of the watchdog timer during debug events.

ROM_WatchdogStallEnable
    // Enables stalling of the watchdog timer during debug events.

ROM_WatchdogUnlock
    // Disables the watchdog timer lock mechanism.

ROM_WatchdogValueGet
    // Gets the current watchdog timer value.
```

C Register Quick Reference

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
System Control Base 0x400F.E000															
DID0, type RO, offset 0x000, reset -															
	VER							CLASS							
MAJOR								MINOR							
PBORCTL, type R/W, offset 0x030, reset 0x0000.7FFD															
														BORIOR	
RIS, type RO, offset 0x050, reset 0x0000.0000															
							MOSCPUPRIS		PLLLRIS					BORRIS	
IMC, type R/W, offset 0x054, reset 0x0000.0000															
							MOSCPUPIM		PLLLIM					BORIM	
MISC, type R/W1C, offset 0x058, reset 0x0000.0000															
							MOSCPUPMIS		PLLLMIS					BORMIS	
RESC, type R/W, offset 0x05C, reset -															
															MOSCFAIL
										WDT1	SW	WDT0	BOR	POR	EXT
RCC, type R/W, offset 0x060, reset 0x0780.3AD1															
				ACG	SYSDIV				USESYSDIV						
		PWRDN		BYPASS	XTAL					OSCSRC				IOSCDIS	MOSCDIS
PLLCFG, type RO, offset 0x064, reset -															
		F									R				
GPIOHBCTL, type R/W, offset 0x06C, reset 0x0000.0000															
											PORTE	PORTD	PORTC	PORTB	PORTA
RCC2, type R/W, offset 0x070, reset 0x0780.6810															
USERCC2	DIV400		SYSDIV2						SYSDIV2LSB						
		PWRDN2		BYPASS2					OSCSRC2						
MOSCCTL, type R/W, offset 0x07C, reset 0x0000.0000															
															CVAL
DSLPCLKCFG, type R/W, offset 0x144, reset 0x0780.0000															
			DSDIVORIDE												
									DSOSCSRC						
PIOSCCAL, type R/W, offset 0x150, reset 0x0000.0000															
UTEN															
						CAL	UPDATE		UT						
PIOSCSTAT, type RO, offset 0x154, reset 0x0000.0040															
									DT						
						RESULT			CT						
DID1, type RO, offset 0x004, reset -															
VER				FAM				PARTNO							
PINCOUNT								TEMP			PKG		ROHS	QUAL	
DC0, type RO, offset 0x008, reset 0x001F.000F															
SRAMSZ															
FLASHSZ															

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
DC1, type RO, offset 0x010, reset -															
			WDT1												ADC0
MINSYSDIV						MAXADC0SPD		MPU	HIB	TEMPSNS	PLL	WDT0	SWO	SWD	JTAG
DC2, type RO, offset 0x014, reset 0x0307.5037															
						COMP1	COMP0						TIMER2	TIMER1	TIMER0
	I2C1		I2C0							SSI1	SSI0		UART2	UART1	UART0
DC3, type RO, offset 0x018, reset 0xBFFF.0FC0															
32KHZ		CCP5	CCP4	CCP3	CCP2	CCP1	CCP0	ADC0AIN7	ADC0AIN6	ADC0AIN5	ADC0AIN4	ADC0AIN3	ADC0AIN2	ADC0AIN1	ADC0AIN0
				C1O	C1PLUS	C1MINUS	C0O	C0PLUS	C0MINUS						
DC4, type RO, offset 0x01C, reset 0x0004.301F															
													PICAL		
		UDMA	ROM								GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
DC5, type RO, offset 0x020, reset 0x0000.0000															
DC6, type RO, offset 0x024, reset 0x0000.0000															
DC7, type RO, offset 0x028, reset 0xFFFF.FFFF															
	DMACH30	DMACH29	DMACH28	DMACH27	DMACH26	DMACH25	DMACH24	DMACH23	DMACH22	DMACH21	DMACH20	DMACH19	DMACH18	DMACH17	DMACH16
DMACH15	DMACH14	DMACH13	DMACH12	DMACH11	DMACH10	DMACH9	DMACH8	DMACH7	DMACH6	DMACH5	DMACH4	DMACH3	DMACH2	DMACH1	DMACH0
DC8, type RO, offset 0x02C, reset 0x0000.00FF															
								ADC0AIN7	ADC0AIN6	ADC0AIN5	ADC0AIN4	ADC0AIN3	ADC0AIN2	ADC0AIN1	ADC0AIN0
DC9, type RO, offset 0x190, reset 0x0000.00FF															
								ADC0DC7	ADC0DC6	ADC0DC5	ADC0DC4	ADC0DC3	ADC0DC2	ADC0DC1	ADC0DC0
NVMSTAT, type RO, offset 0x1A0, reset 0x0000.0001															
															FWB
RCGC0, type R/W, offset 0x100, reset 0x00000040															
			WDT1												ADC0
						MAXADC0SPD			HIB			WDT0			
SCGC0, type R/W, offset 0x110, reset 0x00000040															
			WDT1												ADC0
						MAXADC0SPD			HIB			WDT0			
DCGC0, type R/W, offset 0x120, reset 0x00000040															
			WDT1												ADC0
									HIB			WDT0			
RCGC1, type R/W, offset 0x104, reset 0x00000000															
						COMP1	COMP0						TIMER2	TIMER1	TIMER0
	I2C1		I2C0							SSI1	SSI0		UART2	UART1	UART0
SCGC1, type R/W, offset 0x114, reset 0x00000000															
						COMP1	COMP0						TIMER2	TIMER1	TIMER0
	I2C1		I2C0							SSI1	SSI0		UART2	UART1	UART0
DCGC1, type R/W, offset 0x124, reset 0x00000000															
						COMP1	COMP0						TIMER2	TIMER1	TIMER0
	I2C1		I2C0							SSI1	SSI0		UART2	UART1	UART0
RCGC2, type R/W, offset 0x108, reset 0x00000000															
		UDMA									GPIOE	GPIOD	GPIOC	GPIOB	GPIOA

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCGC2, type R/W, offset 0x118, reset 0x00000000															
		UDMA									GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
DCGC2, type R/W, offset 0x128, reset 0x00000000															
		UDMA									GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
SRCR0, type R/W, offset 0x040, reset 0x00000000															
			WDT1												ADC0
									HIB			WDT0			
SRCR1, type R/W, offset 0x044, reset 0x00000000															
						COMP1	COMP0						TIMER2	TIMER1	TIMER0
	I2C1		I2C0							SSI1	SSI0		UART2	UART1	UART0
SRCR2, type R/W, offset 0x048, reset 0x00000000															
		UDMA									GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Hibernation Module Base 0x400F.C000															
HIBRTCC, type RO, offset 0x000, reset 0x0000.0000															
RTCC															
RTCC															
HIBRTCM0, type R/W, offset 0x004, reset 0xFFFF.FFFF															
RTCM0															
RTCM0															
HIBRTCM1, type R/W, offset 0x008, reset 0xFFFF.FFFF															
RTCM1															
RTCM1															
HIBRTCLD, type R/W, offset 0x00C, reset 0xFFFF.FFFF															
RTCLD															
RTCLD															
HIBCTL, type R/W, offset 0x010, reset 0x8000.0000															
WRC															
							VDD3ON	VABORT	CLK32EN	LOWBATEN	PINWEN	RTCWEN	CLKSEL	HIBREQ	RTCEN
HIBIM, type R/W, offset 0x014, reset 0x0000.0000															
												EXTW	LOWBAT	RTCALT1	RTCALT0
HIBRIS, type RO, offset 0x018, reset 0x0000.0000															
												EXTW	LOWBAT	RTCALT1	RTCALT0
HIBMIS, type RO, offset 0x01C, reset 0x0000.0000															
												EXTW	LOWBAT	RTCALT1	RTCALT0
HIBIC, type R/W1C, offset 0x020, reset 0x0000.0000															
												EXTW	LOWBAT	RTCALT1	RTCALT0
HIBRTCT, type R/W, offset 0x024, reset 0x0000.7FFF															
TRIM															
HIBDATA, type R/W, offset 0x030-0x12C, reset -															
RTD															
RTD															

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
Internal Memory **Flash Memory Registers (Flash Control Offset)** Base 0x400F.D000															
FMA, type R/W, offset 0x000, reset 0x0000.0000															
	OFFSET														
FMD, type R/W, offset 0x004, reset 0x0000.0000															
DATA															
DATA															
FMC, type R/W, offset 0x008, reset 0x0000.0000															
WRKEY															
												COMT	MERASE	ERASE	WRITE
FCRIS, type RO, offset 0x00C, reset 0x0000.0000															
														PRIS	ARIS
FCIM, type R/W, offset 0x010, reset 0x0000.0000															
														PMASK	AMASK
FCMISC, type R/W1C, offset 0x014, reset 0x0000.0000															
														PMISC	AMISC
FMC2, type R/W, offset 0x020, reset 0x0000.0000															
WRKEY															
															WRBUF
FWBVAL, type R/W, offset 0x030, reset 0x0000.0000															
FWB[n]															
FWB[n]															
FWBn, type R/W, offset 0x100 - 0x17C, reset 0x0000.0000															
DATA															
DATA															
FCTL, type R/W, offset 0x0F8, reset 0x0000.0000															
														USDACK	USDREQ
Internal Memory **Memory Registers (System Control Offset)** Base 0x400F.E000															
RMCTL, type R/W1C, offset 0x0F0, reset -															
															BA
RMVER, type RO, offset 0x0F4, reset 0x0505.0400															
CONT								SIZE							
VER								REV							
FMPRE0, type R/W, offset 0x130 and 0x200, reset 0x0000.FFFF															
READ_ENABLE															
READ_ENABLE															
FMPPE0, type R/W, offset 0x134 and 0x400, reset 0x0000.FFFF															
PROG_ENABLE															
PROG_ENABLE															
USER_DBG, type R/W, offset 0x1D0, reset 0xFFFF.FFFE															
NW	DATA														
DATA														DBG1	DBG0

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_REG0, type R/W, offset 0x1E0, reset 0xFFFF.FFFF															
NW	DATA														
DATA															
USER_REG1, type R/W, offset 0x1E4, reset 0xFFFF.FFFF															
NW	DATA														
DATA															
USER_REG2, type R/W, offset 0x1E8, reset 0xFFFF.FFFF															
NW	DATA														
DATA															
USER_REG3, type R/W, offset 0x1EC, reset 0xFFFF.FFFF															
NW	DATA														
DATA															
FMPRE1, type R/W, offset 0x204, reset 0x0000.0000															
READ_ENABLE															
READ_ENABLE															
FMPRE2, type R/W, offset 0x208, reset 0x0000.0000															
READ_ENABLE															
READ_ENABLE															
FMPRE3, type R/W, offset 0x20C, reset 0x0000.0000															
READ_ENABLE															
READ_ENABLE															
FMPPE1, type R/W, offset 0x404, reset 0x0000.0000															
PROG_ENABLE															
PROG_ENABLE															
FMPPE2, type R/W, offset 0x408, reset 0x0000.0000															
PROG_ENABLE															
PROG_ENABLE															
FMPPE3, type R/W, offset 0x40C, reset 0x0000.0000															
PROG_ENABLE															
PROG_ENABLE															
Micro Direct Memory Access (µDMA) **µDMA Channel Control Structure (Offset from Channel Control Table Base)** Base n/a															
DMASRCENDP, type R/W, offset 0x000, reset -															
ADDR															
ADDR															
DMADSTENDP, type R/W, offset 0x004, reset -															
ADDR															
ADDR															
DMACHCTL, type R/W, offset 0x008, reset -															
DSTINC		DSTSIZE		SRCINC		SRCSIZE								ARBSIZE	
ARBSIZE		XFERSIZE										NXTUSEBURST	XFERMODE		
Micro Direct Memory Access (µDMA) **µDMA Registers (Offset from µDMA Base Address)** Base 0x400F.F000															
DMASTAT, type RO, offset 0x000, reset 0x001F.0000															
											DMACHANS				
								STATE							MASTEN
DMACFG, type WO, offset 0x004, reset -															
															MASTEN

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
DMACTLBASE, type R/W, offset 0x008, reset 0x0000.0000															
ADDR															
ADDR															
DMAALTBASE, type RO, offset 0x00C, reset 0x0000.0200															
ADDR															
ADDR															
DMAWAITSTAT, type RO, offset 0x010, reset 0x0000.0000															
WAITREQ[n]															
WAITREQ[n]															
DMASWREQ, type WO, offset 0x014, reset -															
SWREQ[n]															
SWREQ[n]															
DMAUSEBURSTSET, type R/W, offset 0x018, reset 0x0000.0000															
SET[n]															
SET[n]															
DMAUSEBURSTCLR, type WO, offset 0x01C, reset -															
CLR[n]															
CLR[n]															
DMAREQMASKSET, type R/W, offset 0x020, reset 0x0000.0000															
SET[n]															
SET[n]															
DMAREQMASKCLR, type WO, offset 0x024, reset -															
CLR[n]															
CLR[n]															
DMAENASET, type R/W, offset 0x028, reset 0x0000.0000															
SET[n]															
SET[n]															
DMAENACLR, type WO, offset 0x02C, reset -															
CLR[n]															
CLR[n]															
DMAALTSET, type R/W, offset 0x030, reset 0x0000.0000															
SET[n]															
SET[n]															
DMAALTCLR, type WO, offset 0x034, reset -															
CLR[n]															
CLR[n]															
DMAPRIOSET, type R/W, offset 0x038, reset 0x0000.0000															
SET[n]															
SET[n]															
DMAPRIOCLR, type WO, offset 0x03C, reset -															
CLR[n]															
CLR[n]															
DMAERRCLR, type R/W, offset 0x04C, reset 0x0000.0000															
															ERRCLR
DMACHALT, type R/W, offset 0x500, reset 0x0000.0000															
CHALT[n]															
CHALT[n]															
DMAPeriphID0, type RO, offset 0xFE0, reset 0x0000.0030															
								PID0							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAPeriphID1, type RO, offset 0xFE4, reset 0x0000.00B2															
								PID1							
DMAPeriphID2, type RO, offset 0xFE8, reset 0x0000.000B															
								PID2							
DMAPeriphID3, type RO, offset 0xFEC, reset 0x0000.0000															
								PID3							
DMAPeriphID4, type RO, offset 0xFD0, reset 0x0000.0004															
								PID4							
DMAPCellID0, type RO, offset 0xFF0, reset 0x0000.000D															
								CID0							
DMAPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0															
								CID1							
DMAPCellID2, type RO, offset 0xFF8, reset 0x0000.0005															
								CID2							
DMAPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1															
								CID3							
General-Purpose Input/Outputs (GPIOs) GPIO Port A (APB) base: 0x4000.4000 GPIO Port A (AHB) base: 0x4005.8000 GPIO Port B (APB) base: 0x4000.5000 GPIO Port B (AHB) base: 0x4005.9000 GPIO Port C (APB) base: 0x4000.6000 GPIO Port C (AHB) base: 0x4005.A000 GPIO Port D (APB) base: 0x4000.7000 GPIO Port D (AHB) base: 0x4005.B000 GPIO Port E (APB) base: 0x4002.4000 GPIO Port E (AHB) base: 0x4005.C000															
GPIODATA, type R/W, offset 0x000, reset 0x0000.0000															
								DATA							
GPIODIR, type R/W, offset 0x400, reset 0x0000.0000															
								DIR							
GPIOIS, type R/W, offset 0x404, reset 0x0000.0000															
								IS							
GPIOIBE, type R/W, offset 0x408, reset 0x0000.0000															
								IBE							
GPIOIEV, type R/W, offset 0x40C, reset 0x0000.0000															
								IEV							
GPIOIM, type R/W, offset 0x410, reset 0x0000.0000															
								IME							
GPIORIS, type RO, offset 0x414, reset 0x0000.0000															
								RIS							

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
GPIOMIS, type RO, offset 0x418, reset 0x0000.0000															
								MIS							
GPIOICR, type W1C, offset 0x41C, reset 0x0000.0000															
								IC							
GPIOAFSEL, type R/W, offset 0x420, reset -															
								AFSEL							
GPIODR2R, type R/W, offset 0x500, reset 0x0000.00FF															
								DRV2							
GPIODR4R, type R/W, offset 0x504, reset 0x0000.0000															
								DRV4							
GPIODR8R, type R/W, offset 0x508, reset 0x0000.0000															
								DRV8							
GPIOODR, type R/W, offset 0x50C, reset 0x0000.0000															
								ODE							
GPIOPUR, type R/W, offset 0x510, reset -															
								PUE							
GPIOPDR, type R/W, offset 0x514, reset 0x0000.0000															
								PDE							
GPIOSLR, type R/W, offset 0x518, reset 0x0000.0000															
								SRL							
GPIODEN, type R/W, offset 0x51C, reset -															
								DEN							
GPIOLOCK, type R/W, offset 0x520, reset 0x0000.0001															
LOCK															
LOCK															
GPIOCR, type -, offset 0x524, reset -															
								CR							
GPIOAMSEL, type R/W, offset 0x528, reset 0x0000.0000															
												GPIOAMSEL			
GPIOPCTL, type R/W, offset 0x52C, reset -															
PMC7				PMC6				PMC5				PMC4			
PMC3				PMC2				PMC1				PMC0			
GPIOPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000															
								PID4							
GPIOPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000															
								PID5							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000															
								PID6							
GPIOPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000															
								PID7							
GPIOPeriphID0, type RO, offset 0xFE0, reset 0x0000.0061															
								PID0							
GPIOPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000															
								PID1							
GPIOPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018															
								PID2							
GPIOPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001															
								PID3							
GPIOPCellID0, type RO, offset 0xFF0, reset 0x0000.000D															
								CID0							
GPIOPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0															
								CID1							
GPIOPCellID2, type RO, offset 0xFF8, reset 0x0000.0005															
								CID2							
GPIOPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1															
								CID3							
General-Purpose Timers Timer0 base: 0x4003.0000 Timer1 base: 0x4003.1000 Timer2 base: 0x4003.2000															
GPTMCFG, type R/W, offset 0x000, reset 0x0000.0000															
													GPTMCFG		
GPTMTAMR, type R/W, offset 0x004, reset 0x0000.0000															
								TASNAPS	TAWOT	TAMIE	TACDIR	TAAMS	TACMR	TAMR	
GPTMTBMR, type R/W, offset 0x008, reset 0x0000.0000															
								TBSNAPS	TBWOT	TBMIE	TBCDIR	TBAMS	TBCMR	TBMR	
GPTMCTL, type R/W, offset 0x00C, reset 0x0000.0000															
	TBPWML	TBOTE		TBEVENT		TBSTALL	TBEN		TAPWML	TAOTE	RTCEN	TAEVENT		TASTALL	TAEN
GPTMIMR, type R/W, offset 0x018, reset 0x0000.0000															
				TBMIM	CBEIM	CBMIM	TBTOIM				TAMIM	RTCIM	CAEIM	CAMIM	TATOIM
GPTMRIS, type RO, offset 0x01C, reset 0x0000.0000															
				TBMRIS	CBERIS	CBMRIS	TBTORIS				TAMRIS	RTCRIS	CAERIS	CAMRIS	TATORIS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPTMMIS, type RO, offset 0x020, reset 0x0000.0000															
				TBMMIS	CBEMIS	CBMMIS	TBTOMIS				TAMMIS	RTCMIS	CAEMIS	CAMMIS	TATOMIS
GPTMICR, type W1C, offset 0x024, reset 0x0000.0000															
				TBMCINT	CBECINT	CBMCINT	TBTOCINT				TAMCINT	RTCCINT	CAECINT	CAMCINT	TATOCINT
GPTMTAILR, type R/W, offset 0x028, reset 0xFFFF.FFFF															
TAILRH															
TAILRL															
GPTMTBILR, type R/W, offset 0x02C, reset 0x0000.FFFF															
TBILRL															
GPTMTAMATCHR, type R/W, offset 0x030, reset 0xFFFF.FFFF															
TAMRH															
TAMRL															
GPTMTBMATCHR, type R/W, offset 0x034, reset 0x0000.FFFF															
TBMRL															
GPTMTAPR, type R/W, offset 0x038, reset 0x0000.0000															
								TAPSR							
GPTMTBPR, type R/W, offset 0x03C, reset 0x0000.0000															
								TBPSR							
GPTMTAR, type RO, offset 0x048, reset 0xFFFF.FFFF															
TARH															
TARL															
GPTMTBR, type RO, offset 0x04C, reset 0x0000.FFFF															
TBRL															
GPTMTAV, type RO, offset 0x050, reset 0xFFFF.FFFF															
TAVH															
TAVL															
GPTMTBV, type RO, offset 0x054, reset 0x0000.FFFF															
TBVL															
Watchdog Timers WDT0 base: 0x4000.0000 WDT1 base: 0x4000.1000															
WDTLOAD, type R/W, offset 0x000, reset 0xFFFF.FFFF															
WDTLOAD															
WDTLOAD															
WDTVALUE, type RO, offset 0x004, reset 0xFFFF.FFFF															
WDTVALUE															
WDTVALUE															
WDTCTL, type R/W, offset 0x008, reset 0x0000.0000 (WDT0) and 0x8000.0000 (WDT1)															
WRC															
														RESEN	INTEN
WDTICR, type WO, offset 0x00C, reset -															
WDTINTCLR															
WDTINTCLR															

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
WDTRIS, type RO, offset 0x010, reset 0x0000.0000															
															WDTRIS
WDTMIS, type RO, offset 0x014, reset 0x0000.0000															
															WDTMIS
WDTTEST, type R/W, offset 0x418, reset 0x0000.0000															
							STALL								
WDTLOCK, type R/W, offset 0xC00, reset 0x0000.0000															
WDTLOCK															
WDTLOCK															
WDTPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000															
								PID4							
WDTPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000															
								PID5							
WDTPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000															
								PID6							
WDTPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000															
								PID7							
WDTPeriphID0, type RO, offset 0xFE0, reset 0x0000.0005															
								PID0							
WDTPeriphID1, type RO, offset 0xFE4, reset 0x0000.0018															
								PID1							
WDTPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018															
								PID2							
WDTPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001															
								PID3							
WDTPCellID0, type RO, offset 0xFF0, reset 0x0000.000D															
								CID0							
WDTPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0															
								CID1							
WDTPCellID2, type RO, offset 0xFF8, reset 0x0000.0006															
								CID2							
WDTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1															
								CID3							
Analog-to-Digital Converter (ADC) Base 0x4003.8000															
ADCACTSS, type R/W, offset 0x000, reset 0x0000.0000															
												ASEN3	ASEN2	ASEN1	ASEN0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCRIS, type RO, offset 0x004, reset 0x0000.0000															
															INRDC
												INR3	INR2	INR1	INR0
ADCIM, type R/W, offset 0x008, reset 0x0000.0000															
												DCONSS3	DCONSS2	DCONSS1	DCONSS0
												MASK3	MASK2	MASK1	MASK0
ADCISC, type R/W1C, offset 0x00C, reset 0x0000.0000															
												DCINSS3	DCINSS2	DCINSS1	DCINSS0
												IN3	IN2	IN1	IN0
ADCOSTAT, type R/W1C, offset 0x010, reset 0x0000.0000															
												OV3	OV2	OV1	OV0
ADCEMUX, type R/W, offset 0x014, reset 0x0000.0000															
EM3				EM2				EM1				EM0			
ADCUSTAT, type R/W1C, offset 0x018, reset 0x0000.0000															
												UV3	UV2	UV1	UV0
ADCSSPRI, type R/W, offset 0x020, reset 0x0000.3210															
		SS3				SS2				SS1				SS0	
ADCPSSI, type WO, offset 0x028, reset -															
												SS3	SS2	SS1	SS0
ADCSAC, type R/W, offset 0x030, reset 0x0000.0000															
													AVG		
ADCDCISC, type R/W1C, offset 0x034, reset 0x0000.0000															
								DCINT7	DCINT6	DCINT5	DCINT4	DCINT3	DCINT2	DCINT1	DCINT0
ADCCTL, type R/W, offset 0x038, reset 0x0000.0000															
															VREF
ADCSSMUX0, type R/W, offset 0x040, reset 0x0000.0000															
	MUX7				MUX6				MUX5				MUX4		
	MUX3				MUX2				MUX1				MUX0		
ADCSSCTL0, type R/W, offset 0x044, reset 0x0000.0000															
TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
ADCSSFIFO0, type RO, offset 0x048, reset 0x0000.0000															
						DATA									
ADCSSFIFO1, type RO, offset 0x068, reset 0x0000.0000															
						DATA									
ADCSSFIFO2, type RO, offset 0x088, reset 0x0000.0000															
						DATA									
ADCSSFIFO3, type RO, offset 0x0A8, reset 0x0000.0000															
						DATA									

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCSSFSTAT0, type RO, offset 0x04C, reset 0x0000.0100															
			FULL				EMPTY	HPTR				TPTR			
ADCSSFSTAT1, type RO, offset 0x06C, reset 0x0000.0100															
			FULL				EMPTY	HPTR				TPTR			
ADCSSFSTAT2, type RO, offset 0x08C, reset 0x0000.0100															
			FULL				EMPTY	HPTR				TPTR			
ADCSSFSTAT3, type RO, offset 0x0AC, reset 0x0000.0100															
			FULL				EMPTY	HPTR				TPTR			
ADCSSOP0, type R/W, offset 0x050, reset 0x0000.0000															
			S7DCOP				S6DCOP				S5DCOP				S4DCOP
			S3DCOP				S2DCOP				S1DCOP				S0DCOP
ADCSSDC0, type R/W, offset 0x054, reset 0x0000.0000															
S7DCSEL				S6DCSEL				S5DCSEL				S4DCSEL			
S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL			
ADCSSMUX1, type R/W, offset 0x060, reset 0x0000.0000															
	MUX3				MUX2				MUX1				MUX0		
ADCSSMUX2, type R/W, offset 0x080, reset 0x0000.0000															
	MUX3				MUX2				MUX1				MUX0		
ADCSSCTL1, type R/W, offset 0x064, reset 0x0000.0000															
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
ADCSSCTL2, type R/W, offset 0x084, reset 0x0000.0000															
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
ADCSSOP1, type R/W, offset 0x070, reset 0x0000.0000															
			S3DCOP				S2DCOP				S1DCOP				S0DCOP
ADCSSOP2, type R/W, offset 0x090, reset 0x0000.0000															
			S3DCOP				S2DCOP				S1DCOP				S0DCOP
ADCSSDC1, type R/W, offset 0x074, reset 0x0000.0000															
S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL			
ADCSSDC2, type R/W, offset 0x094, reset 0x0000.0000															
S3DCSEL				S2DCSEL				S1DCSEL				S0DCSEL			
ADCSSMUX3, type R/W, offset 0x0A0, reset 0x0000.0000															
													MUX0		
ADCSSCTL3, type R/W, offset 0x0A4, reset 0x0000.0002															
												TS0	IE0	END0	D0
ADCSSOP3, type R/W, offset 0x0B0, reset 0x0000.0000															
															S0DCOP

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
ADCSSDC3, type R/W, offset 0x0B4, reset 0x0000.0000															
												S0DCSEL			
ADCDCRIC, type R/W, offset 0xD00, reset 0x0000.0000															
								DCTRIG7	DCTRIG6	DCTRIG5	DCTRIG4	DCTRIG3	DCTRIG2	DCTRIG1	DCTRIG0
								DCINT7	DCINT6	DCINT5	DCINT4	DCINT3	DCINT2	DCINT1	DCINT0
ADCDCCTL0, type R/W, offset 0xE00, reset 0x0000.0000															
			CTE	CTC		CTM					CIE	CIC		CIM	
ADCDCCTL1, type R/W, offset 0xE04, reset 0x0000.0000															
			CTE	CTC		CTM					CIE	CIC		CIM	
ADCDCCTL2, type R/W, offset 0xE08, reset 0x0000.0000															
			CTE	CTC		CTM					CIE	CIC		CIM	
ADCDCCTL3, type R/W, offset 0xE0C, reset 0x0000.0000															
			CTE	CTC		CTM					CIE	CIC		CIM	
ADCDCCTL4, type R/W, offset 0xE10, reset 0x0000.0000															
			CTE	CTC		CTM					CIE	CIC		CIM	
ADCDCCTL5, type R/W, offset 0xE14, reset 0x0000.0000															
			CTE	CTC		CTM					CIE	CIC		CIM	
ADCDCCTL6, type R/W, offset 0xE18, reset 0x0000.0000															
			CTE	CTC		CTM					CIE	CIC		CIM	
ADCDCCTL7, type R/W, offset 0xE1C, reset 0x0000.0000															
			CTE	CTC		CTM					CIE	CIC		CIM	
ADCDCCMP0, type R/W, offset 0xE40, reset 0x0000.0000															
						COMP1									
						COMP0									
ADCDCCMP1, type R/W, offset 0xE44, reset 0x0000.0000															
						COMP1									
						COMP0									
ADCDCCMP2, type R/W, offset 0xE48, reset 0x0000.0000															
						COMP1									
						COMP0									
ADCDCCMP3, type R/W, offset 0xE4C, reset 0x0000.0000															
						COMP1									
						COMP0									
ADCDCCMP4, type R/W, offset 0xE50, reset 0x0000.0000															
						COMP1									
						COMP0									
ADCDCCMP5, type R/W, offset 0xE54, reset 0x0000.0000															
						COMP1									
						COMP0									
ADCDCCMP6, type R/W, offset 0xE58, reset 0x0000.0000															
						COMP1									
						COMP0									

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCDCCMP7, type R/W, offset 0xE5C, reset 0x0000.0000															
						COMP1									
						COMP0									
Universal Asynchronous Receivers/Transmitters (UARTs) UART0 base: 0x4000.C000 UART1 base: 0x4000.D000 UART2 base: 0x4000.E000															
UARTDR, type R/W, offset 0x000, reset 0x0000.0000															
				OE	BE	PE	FE	DATA							
UARTRSR/UARTECR, type RO, offset 0x004, reset 0x0000.0000 (Read-Only Status Register)															
												OE	BE	PE	FE
UARTRSR/UARTECR, type WO, offset 0x004, reset 0x0000.0000 (Write-Only Error Clear Register)															
								DATA							
UARTFR, type RO, offset 0x018, reset 0x0000.0090															
								TXFE	RXFF	TXFF	RXFE	BUSY			
UARTILPR, type R/W, offset 0x020, reset 0x0000.0000															
								ILPDVSR							
UARTIBRD, type R/W, offset 0x024, reset 0x0000.0000															
DIVINT															
UARTFBRD, type R/W, offset 0x028, reset 0x0000.0000															
										DIVFRAC					
UARTLCRH, type R/W, offset 0x02C, reset 0x0000.0000															
								SPS	WLEN		FEN	STP2	EPS	PEN	BRK
UARTCTL, type R/W, offset 0x030, reset 0x0000.0300															
						RXE	TXE	LBE	LIN	HSE	EOT	SMART	SIRLP	SIREN	UARTEN
UARTIFLS, type R/W, offset 0x034, reset 0x0000.0012															
										RXIFLSEL			TXIFLSEL		
UARTIM, type R/W, offset 0x038, reset 0x0000.0000															
LME5IM	LME1IM	LMSBIM			OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM	DSRIM	DCDIM	CTSIM	RIIM
UARTRIS, type RO, offset 0x03C, reset 0x0000.000F															
LME5RIS	LME1RIS	LMSBRIS			OERIS	BERIS	PERIS	FERIS	RTRIS	TXRIS	RXRIS	DSRRIS	DCDRIS	CTSRIS	RIRIS
UARTMIS, type RO, offset 0x040, reset 0x0000.0000															
LME5MIS	LME1MIS	LMSBMIS			OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS	DSRMIS	DCDMIS	CTSMIS	RIMIS
UARTICR, type W1C, offset 0x044, reset 0x0000.0000															
LME5MIC	LME1MIC	LMSBMIC			OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC	DSRMIC	DCDMIC	CTSMIC	RIMIC
UARTDMACTL, type R/W, offset 0x048, reset 0x0000.0000															
													DMAERR	TXDMAE	RXDMAE

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
UARTLCTL, type R/W, offset 0x090, reset 0x0000.0000															
										BLEN					MASTER
UARTLSS, type RO, offset 0x094, reset 0x0000.0000															
TSS															
UARTLTIM, type RO, offset 0x098, reset 0x0000.0000															
TIMER															
UARTPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000															
								PID4							
UARTPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000															
								PID5							
UARTPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000															
								PID6							
UARTPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000															
								PID7							
UARTPeriphID0, type RO, offset 0xFE0, reset 0x0000.0060															
								PID0							
UARTPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000															
								PID1							
UARTPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018															
								PID2							
UARTPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001															
								PID3							
UARTPCellID0, type RO, offset 0xFF0, reset 0x0000.000D															
								CID0							
UARTPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0															
								CID1							
UARTPCellID2, type RO, offset 0xFF8, reset 0x0000.0005															
								CID2							
UARTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1															
								CID3							

Synchronous Serial Interface (SSI)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000

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
SSICR0, type R/W, offset 0x000, reset 0x0000.0000															
SCR								SPH	SPO	FRF		DSS			

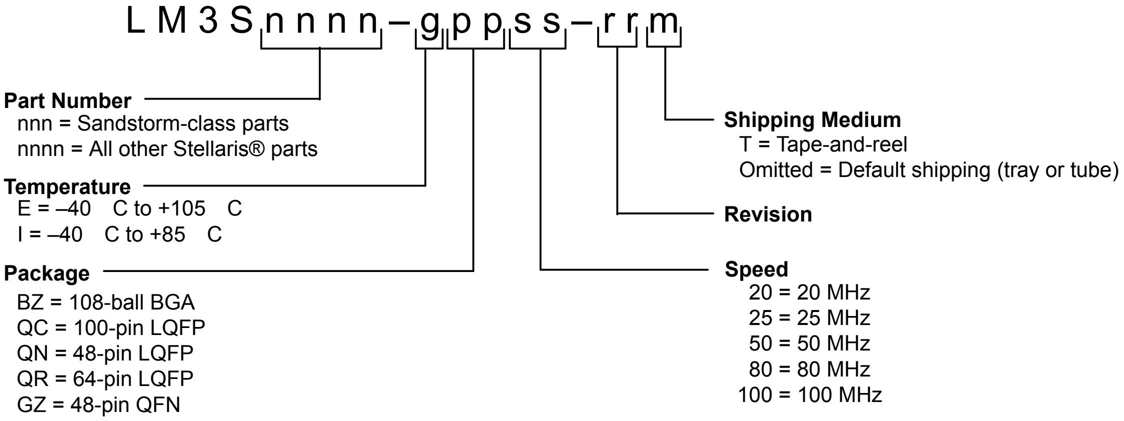
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSICR1, type R/W, offset 0x004, reset 0x0000.0000															
											EOT	SOD	MS	SSE	LBM
SSIDR, type R/W, offset 0x008, reset 0x0000.0000															
DATA															
SSISR, type RO, offset 0x00C, reset 0x0000.0003															
											BSY	RFF	RNE	TNF	TFE
SSICPSR, type R/W, offset 0x010, reset 0x0000.0000															
								CPSDVSR							
SSIIM, type R/W, offset 0x014, reset 0x0000.0000															
												TXIM	RXIM	RTIM	RORIM
SSIRIS, type RO, offset 0x018, reset 0x0000.0008															
												TXRIS	RXRIS	RTRIS	RORRIS
SSIMIS, type RO, offset 0x01C, reset 0x0000.0000															
												TXMIS	RXMIS	RTMIS	RORMIS
SSIICR, type W1C, offset 0x020, reset 0x0000.0000															
														RTIC	RORIC
SSIDMACTL, type R/W, offset 0x024, reset 0x0000.0000															
														TXDMAE	RXDMAE
SSIPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000															
								PID4							
SSIPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000															
								PID5							
SSIPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000															
								PID6							
SSIPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000															
								PID7							
SSIPeriphID0, type RO, offset 0xFE0, reset 0x0000.0022															
								PID0							
SSIPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000															
								PID1							
SSIPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018															
								PID2							
SSIPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001															
								PID3							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSIPCellID0, type RO, offset 0xFF0, reset 0x0000.000D															
								CID0							
SSIPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0															
								CID1							
SSIPCellID2, type RO, offset 0xFF8, reset 0x0000.0005															
								CID2							
SSIPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1															
								CID3							
Inter-Integrated Circuit (I^{2}C) Interface **I^{2}C Master** I2C Master 0 base: 0x4002.0000 I2C Master 1 base: 0x4002.1000															
I2CMSA, type R/W, offset 0x000, reset 0x0000.0000															
								SA							R/S
I2CMCS, type RO, offset 0x004, reset 0x0000.0000 (Read-Only Status Register)															
									BUSBSY	IDLE	ARBLST	DATACK	ADRACK	ERROR	BUSY
I2CMCS, type WO, offset 0x004, reset 0x0000.0000 (Write-Only Control Register)															
												ACK	STOP	START	RUN
I2CMDR, type R/W, offset 0x008, reset 0x0000.0000															
								DATA							
I2CMTPR, type R/W, offset 0x00C, reset 0x0000.0001															
								TPR							
I2CMIMR, type R/W, offset 0x010, reset 0x0000.0000															
															IM
I2CMRIS, type RO, offset 0x014, reset 0x0000.0000															
															RIS
I2CMMIS, type RO, offset 0x018, reset 0x0000.0000															
															MIS
I2CMICR, type WO, offset 0x01C, reset 0x0000.0000															
															IC
I2CMCR, type R/W, offset 0x020, reset 0x0000.0000															
										SFE	MFE				LPBK
Inter-Integrated Circuit (I^{2}C) Interface **I^{2}C Slave** I2C Slave 0 base: 0x4002.0800 I2C Slave 1 base: 0x4002.1800															
I2CSOAR, type R/W, offset 0x000, reset 0x0000.0000															
									OAR						

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
I2CSCSR, type RO, offset 0x004, reset 0x0000.0000 (Read-Only Status Register)															
													FBR	TREQ	RREQ
I2CSCSR, type WO, offset 0x004, reset 0x0000.0000 (Write-Only Control Register)															
															DA
I2CSDR, type R/W, offset 0x008, reset 0x0000.0000															
								DATA							
I2CSIMR, type R/W, offset 0x00C, reset 0x0000.0000															
													STOPIM	STARTIM	DATAIM
I2CSRIS, type RO, offset 0x010, reset 0x0000.0000															
													STOPRIS	STARTRIS	DATARIS
I2CSMIS, type RO, offset 0x014, reset 0x0000.0000															
													STOPMIS	STARTMIS	DATAMIS
I2CSICR, type WO, offset 0x018, reset 0x0000.0000															
													STOPIC	STARTIC	DATAIC
Analog Comparators Base 0x4003.C000															
ACMIS, type R/W1C, offset 0x000, reset 0x0000.0000															
														IN1	IN0
ACRIS, type RO, offset 0x004, reset 0x0000.0000															
														IN1	IN0
ACINTEN, type R/W, offset 0x008, reset 0x0000.0000															
														IN1	IN0
ACREFCTL, type R/W, offset 0x010, reset 0x0000.0000															
						EN	RNG					VREF			
ACSTAT0, type RO, offset 0x020, reset 0x0000.0000															
														OVAL	
ACSTAT1, type RO, offset 0x040, reset 0x0000.0000															
														OVAL	
ACCTL0, type R/W, offset 0x024, reset 0x0000.0000															
				TOEN	ASRCP			TSLVAL	TSEN		ISLVAL	ISEN		CINV	
ACCTL1, type R/W, offset 0x044, reset 0x0000.0000															
				TOEN	ASRCP			TSLVAL	TSEN		ISLVAL	ISEN		CINV	

D Ordering and Contact Information

D.1 Ordering Information



Part Number
nnn = Sandstorm-class parts
nnnn = All other Stellaris® parts

Temperature
E = –40 C to +105 C
I = –40 C to +85 C

Package
BZ = 108-ball BGA
QC = 100-pin LQFP
QN = 48-pin LQFP
QR = 64-pin LQFP
GZ = 48-pin QFN

Shipping Medium
T = Tape-and-reel
Omitted = Default shipping (tray or tube)

Revision

Speed
20 = 20 MHz
25 = 25 MHz
50 = 50 MHz
80 = 80 MHz
100 = 100 MHz

Table D-1. Part Ordering Information

Orderable Part Number	Description
LM3S1W16-IQR50-C0	Stellaris® LM3S1W16 Microcontroller Industrial Temperature 64-pin LQFP
LM3S1W16-IQR50-C0T	Stellaris® LM3S1W16 Microcontroller Industrial Temperature 64-pin LQFP Tape-and-reel

D.2 Part Markings

The Stellaris® microcontrollers are marked with an identifying number. This code contains the following information:

- The first line indicates the part number. In the example figure below, this is the LM3S6965.
- The first seven characters in the second line indicate the temperature, package, speed, and revision. In the example figure below, this is an Industrial temperature (I), 100-pin LQFP package (QC), 50-MHz (50), revision A2 (A2) device.
- The remaining characters contain internal tracking numbers.

D.3 Kits

The Stellaris® Family provides the hardware and software tools that engineers need to begin development quickly.

- Reference Design Kits accelerate product development by providing ready-to-run hardware and comprehensive documentation including hardware design files:

  http://www.luminarymicro.com/products/reference_design_kits/

- Evaluation Kits provide a low-cost and effective means of evaluating Stellaris® microcontrollers before purchase:

  http://www.luminarymicro.com/products/kits.html

- Development Kits provide you with all the tools you need to develop and prototype embedded applications right out of the box:

  http://www.luminarymicro.com/products/development_kits.html

See the website for the latest tools available, or ask your distributor.

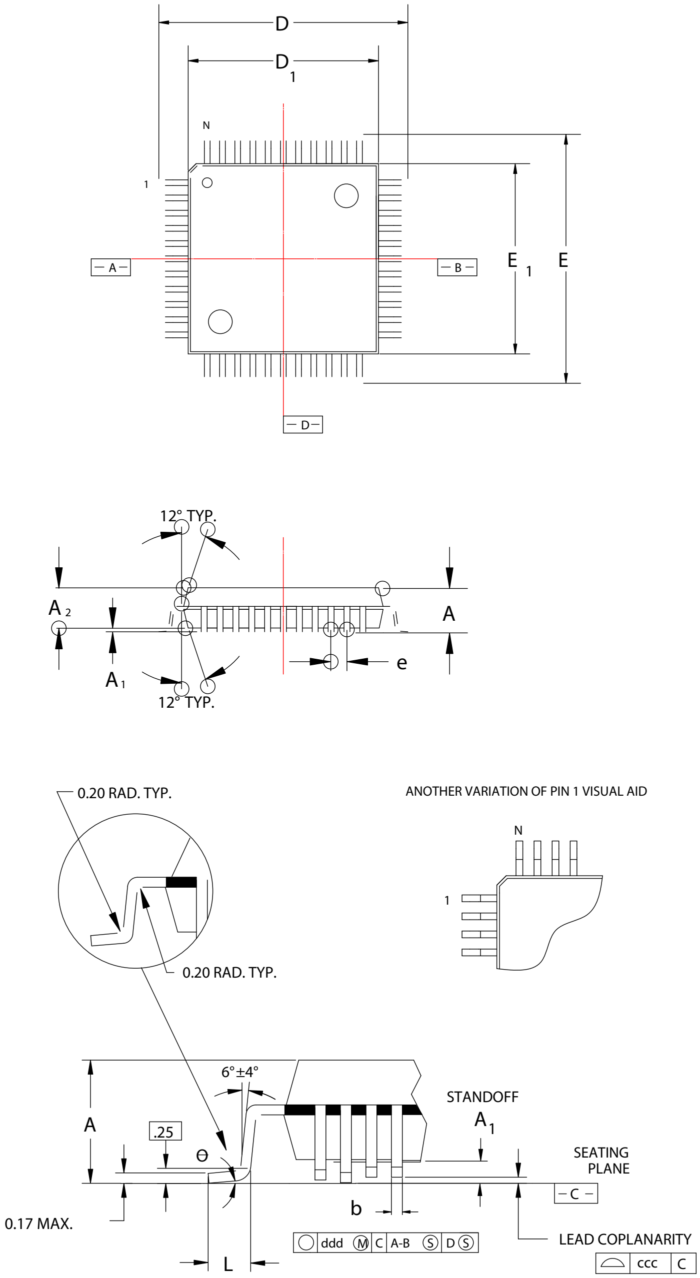
D.4 Support Information

For support on Stellaris® products, contact:

support_lmi@ti.com

E Package Information

Figure E-1. 64-Pin LQFP Package



Note: The following notes apply to the package drawing.

1. All dimensions shown in mm.

2. Dimensions shown are nominal with tolerances indicated.
3. Foot length 'L' is measured at gage plane 0.25 mm above seating plane.
4. L/F: Eftec 64T Cu or equivalent, 0.127mm (0.005") thick.

Body +2.00 mm Footprint, 1.4 mm package thickness		
Symbols	Leads	64L
A	Max.	1.60
A_1	-	0.05 Min./0.15 Max.
A_2	±0.05	1.40
D	±0.20	12.00
D_1	±0.10	10.00
E	±0.20	12.00
E_1	±0.10	10.00
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		BCD

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Broadband	www.ti.com/broadband
DSP	dsp.ti.com	Digital Control	www.ti.com/digitalcontrol
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Military	www.ti.com/military
Logic	logic.ti.com	Optical Networking	www.ti.com/opticalnetwork
Power Mgmt	power.ti.com	Security	www.ti.com/security
Microcontrollers	microcontroller.ti.com	Telephony	www.ti.com/telephony
RFID	www.ti-rfid.com	Video & Imaging	www.ti.com/video
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265