LUMINARY MICRO™

LM3S600 Microcontroller

DATA SHEET

Legal Disclaimers and Trademark Information

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

Table of Contents

List of Figures

List of Tables

List of Registers

About This Document

This data sheet provides reference information for the LM3S600 microcontroller, describing the functional blocks of the system-on-chip (SoC) device designed around the ARM® Cortex™-M3 core.

Audience

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

About This Manual

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

Related Documents

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

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

The following related documents are also referenced:

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

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

Documentation Conventions

This document uses the conventions shown in Table 1 on page 15.

Table 1. Documentation Conventions

Notation	Meaning
General Register Notation	
REGISTER	APB registers are indicated in uppercase bold. For example, **PBORCTL** is the Power-On and Brown-Out Reset Control register. If a register name contains a lowercase n, it represents more than one register. For example, **SRCRn** represents any (or all) of the three Software Reset Control registers: **SRCR0, SRCR1** , and **SRCR2**.
bit	A single bit in a register.
bit field	Two or more consecutive and related bits.
offset 0x*nnn*	A hexadecimal increment to a register's address, relative to that module's base address as specified in “Memory Map” on page 34.
Register *N*	Registers are numbered consecutively throughout the document to aid in referencing them. The register number has no meaning to software.

Notation	Meaning
reserved	Register bits marked *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.
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.
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

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

The LM3S600 microcontroller is targeted for industrial applications, including test and measurement equipment, factory automation, HVAC and building control, motion control, medical instrumentation, fire and security, and power/energy.

In addition, the LM3S600 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 LM3S600 microcontroller is code-compatible to all members of the extensive Stellaris® family; providing flexibility to fit our customers' precise needs.

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

1.1 Product Features

The LM3S600 microcontroller includes the following product features:

- 32-Bit RISC Performance
  - 32-bit ARM® Cortex™-M3 v7M architecture optimized for small-footprint embedded applications
  - System timer (SysTick), providing a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism
  - Thumb®-compatible Thumb-2-only instruction set processor core for high code density
  - 50-MHz operation
  - Hardware-division and single-cycle-multiplication
  - Integrated Nested Vectored Interrupt Controller (NVIC) providing deterministic interrupt handling
  - 21 interrupts with eight priority levels
  - Memory protection unit (MPU), providing a privileged mode for protected operating system functionality
  - Unaligned data access, enabling data to be efficiently packed into memory
  - Atomic bit manipulation (bit-banding), delivering maximum memory utilization and streamlined peripheral control
- Internal Memory

- 32 KB single-cycle flash
  - User-managed flash block protection on a 2-KB block basis
  - User-managed flash data programming
  - User-defined and managed flash-protection block
- 8 KB single-cycle SRAM

■ General-Purpose Timers

- Three General-Purpose Timer Modules (GPTM), each of which provides two 16-bit timer/counters. Each GPTM can be configured to operate independently as timers or event counters as a single 32-bit timer, as one 32-bit Real-Time Clock (RTC) to event capture, or for Pulse Width Modulation (PWM)
- 32-bit Timer modes
  - Programmable one-shot timer
  - Programmable periodic timer
  - Real-Time Clock when using an external 32.768-KHz clock as the input
  - User-enabled stalling in periodic and one-shot mode when the controller asserts the CPU Halt flag during debug
- 16-bit Timer modes
  - General-purpose timer function with an 8-bit prescaler
  - Programmable one-shot timer
  - Programmable periodic timer
  - User-enabled stalling when the controller asserts CPU Halt flag during debug
- 16-bit Input Capture modes
  - Input edge count capture
  - Input edge time capture
- 16-bit PWM mode
  - Simple PWM mode with software-programmable output inversion of the PWM signal

■ ARM FiRM-compliant Watchdog Timer

- 32-bit down counter with a programmable load register
- Separate watchdog clock with an enable
- Programmable interrupt generation logic with interrupt masking
- Lock register protection from runaway software

- Reset generation logic with an enable/disable
- User-enabled stalling when the controller asserts the CPU Halt flag during debug

- Synchronous Serial Interface (SSI)
  - Master or slave operation
  - Programmable clock bit rate and prescale
  - Separate transmit and receive FIFOs, 16 bits wide, 8 locations deep
  - Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces
  - Programmable data frame size from 4 to 16 bits
  - Internal loopback test mode for diagnostic/debug testing
- UART
  - Two fully programmable 16C550-type UARTs
  - Separate 16x8 transmit (TX) and 16x12 receive (RX) FIFOs to reduce CPU interrupt service loading
  - Programmable baud-rate generator with fractional divider
  - Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
  - FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
  - Standard asynchronous communication bits for start, stop, and parity
  - False-start-bit detection
  - Line-break generation and detection
- Analog Comparators
  - Three independent integrated analog comparators
  - Configurable for output to: drive an output pin or generate an interrupt
  - Compare external pin input to external pin input or to internal programmable voltage reference
- I^2C
  - Master and slave receive and transmit operation with transmission speed up to 100 Kbps in Standard mode and 400 Kbps in Fast mode
  - Interrupt generation
  - Master with arbitration and clock synchronization, multimaster support, and 7-bit addressing mode

- GPIOs
  - 8-36 GPIOs, depending on configuration
  - 5-V-tolerant input/outputs
  - Programmable interrupt generation as either edge-triggered or level-sensitive
  - Bit masking in both read and write operations through address lines
  - Programmable control for GPIO pad configuration:
    - Weak pull-up or pull-down resistors
    - 2-mA, 4-mA, and 8-mA pad drive
    - Slew rate control for the 8-mA drive
    - Open drain enables
    - Digital input enables
- Power
  - On-chip Low Drop-Out (LDO) voltage regulator, with programmable output user-adjustable from 2.25 V to 2.75 V
  - Low-power options on controller: Sleep and Deep-sleep modes
  - Low-power options for peripherals: software controls shutdown of individual peripherals
  - User-enabled LDO unregulated voltage detection and automatic reset
  - 3.3-V supply brown-out detection and reporting via interrupt or reset
- Flexible Reset Sources
  - Power-on reset (POR)
  - Reset pin assertion
  - Brown-out (BOR) detector alerts to system power drops
  - Software reset
  - Watchdog timer reset
  - Internal low drop-out (LDO) regulator output goes unregulated
- Additional Features
  - Six reset sources
  - Programmable clock source control
  - Clock gating to individual peripherals for power savings

- IEEE 1149.1-1990 compliant Test Access Port (TAP) controller
- Debug access via JTAG and Serial Wire interfaces
- Full JTAG boundary scan

- Industrial-range 48-pin RoHS-compliant LQFP package

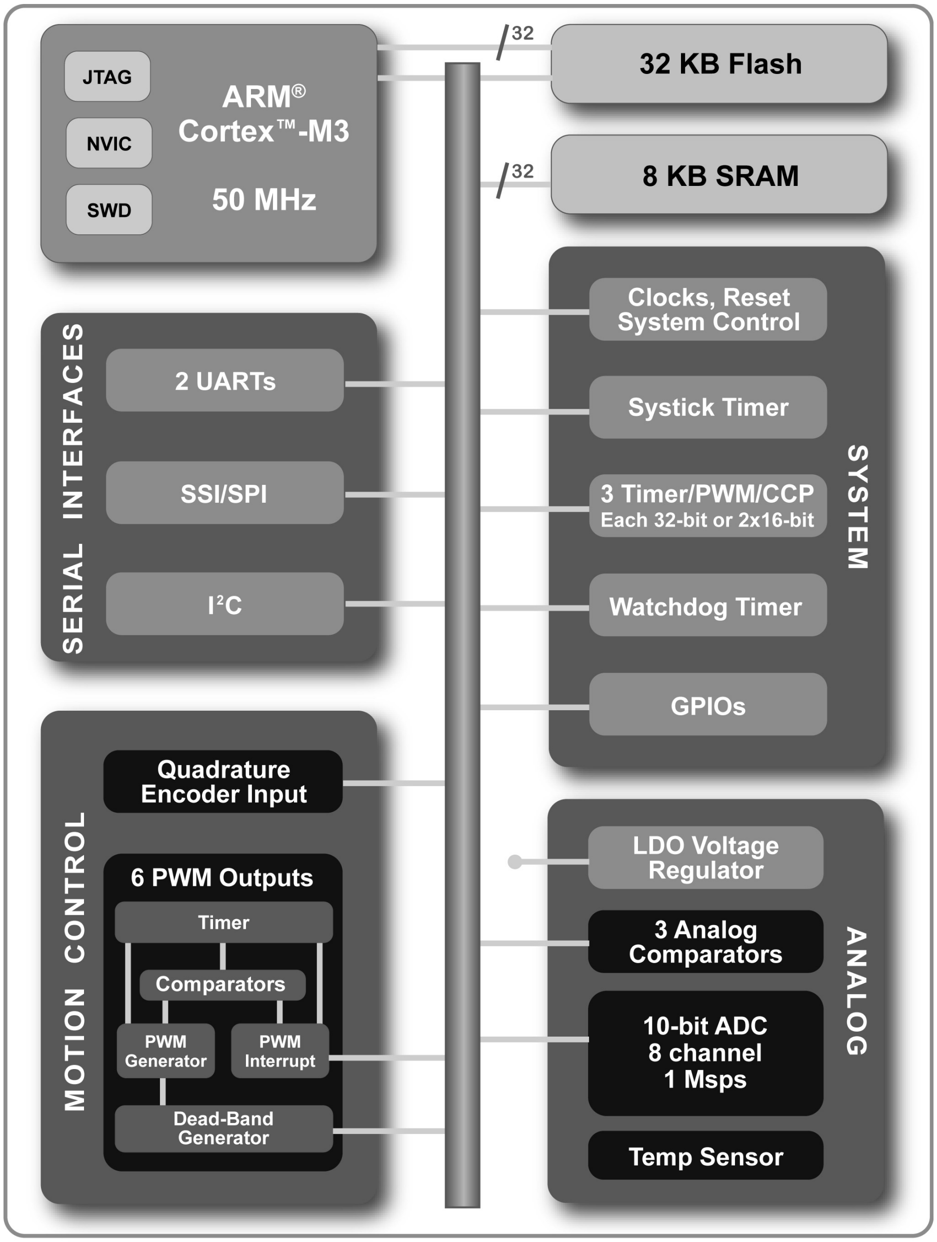
1.2 Target Applications

- Factory automation and control
- Industrial control power devices
- Building and home automation
- Stepper motors
- Brushless DC motors
- AC induction motors

1.3 High-Level Block Diagram

Figure 1-1 on page 22 represents the full set of features in the Stellaris® 600 series of devices; not all features may be available on the LM3S600 microcontroller.

Figure 1-1. Stellaris® 600 Series High-Level Block Diagram



1.4 Functional Overview

The following sections provide an overview of the features of the LM3S600 microcontroller. The page number in parenthesis indicates where that feature is discussed in detail. Ordering and support information can be found in “Ordering and Contact Information” on page 377.

1.4.1 ARM Cortex™-M3

1.4.1.1 Processor Core (see page 28)

All members of the Stellaris® product family, including the LM3S600 microcontroller, are designed around an ARM Cortex™-M3 processor core. The ARM Cortex-M3 processor provides the core for a high-performance, low-cost platform that meets the needs of minimal memory implementation, reduced pin count, and low-power consumption, while delivering outstanding computational performance and exceptional system response to interrupts.

"ARM Cortex-M3 Processor Core" on page 28 provides an overview of the ARM core; the core is detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

1.4.1.2 System Timer (SysTick)

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

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

1.4.1.3 Nested Vectored Interrupt Controller (NVIC)

The LM3S600 controller includes the ARM Nested Vectored Interrupt Controller (NVIC) on the ARM Cortex-M3 core. The NVIC and Cortex-M3 prioritize and handle all exceptions. All exceptions are handled in Handler Mode. The processor state is automatically stored to the stack on an exception, and automatically restored from the stack at the end of the Interrupt Service Routine (ISR). The vector is fetched in parallel to the state saving, which enables efficient interrupt entry. The processor supports tail-chaining, which enables back-to-back interrupts to be performed without the overhead of state saving and restoration. Software can set eight priority levels on 7 exceptions (system handlers) and 21 interrupts.

"Interrupts" on page 36 provides an overview of the NVIC controller and the interrupt map. Exceptions and interrupts are detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

1.4.2 Motor Control Peripherals

To enhance motor control, the LM3S600 controller features Pulse Width Modulation (PWM) outputs.

1.4.2.1 PWM (see page 163)

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

On the LM3S600, PWM motion control functionality can be achieved through the motion control features of the general-purpose timers (using the CCP pins).

CCP Pins (see page 163)

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

1.4.3 Analog Peripherals

For support of analog signals, the LM3S600 microcontroller offers three analog comparators.

1.4.3.1 Analog Comparators (see page 326)

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

The LM3S600 microcontroller provides three independent integrated analog comparators that can be configured to drive an output or generate an interrupt .

A comparator can compare a test voltage against any one of these voltages:

- An individual external reference voltage
- A shared single external reference voltage
- A shared internal reference voltage

The comparator can provide its output to a device pin, acting as a replacement for an analog comparator on the board, or it can be used to signal the application via interrupts to cause it to start capturing a sample sequence.

1.4.4 Serial Communications Peripherals

The LM3S600 controller supports both asynchronous and synchronous serial communications with:

- Two fully programmable 16C550-type UARTs
- One SSI module
- One I^2C module

1.4.4.1 UART (see page 216)

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 LM3S600 controller includes two fully programmable 16C550-type UARTs that support data transfer speeds up to 460.8 Kbps. (Although similar in functionality to a 16C550 UART, it is not register-compatible.)

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

1.4.4.2 SSI (see page 254)

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

The LM3S600 controller includes one SSI module that provides the functionality for synchronous serial communications with peripheral devices, and can be configured to use the Freescale SPI, MICROWIRE, or TI synchronous serial interface frame formats. The size of the data frame is also configurable, and can be set between 4 and 16 bits, inclusive.

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

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

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

1.4.4.3 I^2C (see page 291)

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).

The I^2C bus 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 LM3S600 controller includes one I^2C module that provides the ability to communicate to other IC devices over an I^2C bus. The I^2C bus supports devices that can both transmit and receive (write and read) data.

Devices on the I^2C bus can be designated as either a master or a slave. The I^2C module supports both sending and receiving data as either a master or a slave, and also supports the simultaneous operation as both a master and a slave. The four I^2C modes are: Master Transmit, Master Receive, Slave Transmit, and Slave Receive.

A Stellaris® I^2C module can operate at two speeds: Standard (100 Kbps) and Fast (400 Kbps).

Both the I^2C master and slave can generate interrupts. The I^2C master generates interrupts when a transmit or receive operation completes (or aborts due to an error). The I^2C slave generates interrupts when data has been sent or requested by a master.

1.4.5 System Peripherals

1.4.5.1 Programmable GPIOs (see page 119)

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

The Stellaris® GPIO module is composed 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 8-36 programmable input/output pins. The number of GPIOs available depends on the peripherals being used (see “Signal Tables” on page 340 for the signals available to each GPIO pin).

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

1.4.5.2 Three Programmable Timers (see page 157)

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. Each GPTM block provides two 16-bit timer/counters that can be configured to operate independently as timers or event counters, or configured to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC).

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

1.4.5.3 Watchdog Timer (see page 193)

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

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

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

1.4.6 Memory Peripherals

The LM3S600 controller offers both single-cycle SRAM and single-cycle Flash memory.

1.4.6.1 SRAM (see page 103)

The LM3S600 static random access memory (SRAM) controller supports 8 KB SRAM. The internal SRAM of the Stellaris® devices is located at offset 0x0000.0000 of the device memory map. To reduce the number of time-consuming read-modify-write (RMW) operations, ARM has introduced *bit-banding* technology in the new Cortex-M3 processor. With a bit-band-enabled processor, certain regions in the memory map (SRAM and peripheral space) can use address aliases to access individual bits in a single, atomic operation.

1.4.6.2 Flash (see page 104)

The LM3S600 Flash controller supports 32 KB of flash memory. The flash is organized as a set of 1-KB blocks that can be individually erased. Erasing a block causes the entire contents of the block to be reset to all 1s. These blocks are paired into a set of 2-KB blocks that can be individually protected. The blocks can be marked as read-only or execute-only, providing different levels of code protection. Read-only blocks cannot be erased or programmed, protecting the contents of those blocks from being modified. Execute-only blocks cannot be erased or programmed, and can only be read by the controller instruction fetch mechanism, protecting the contents of those blocks from being read by either the controller or by a debugger.

1.4.7 Additional Features

1.4.7.1 Memory Map (see page 34)

A memory map lists the location of instructions and data in memory. The memory map for the LM3S600 controller can be found in "Memory Map" on page 34. Register addresses are given as a hexadecimal increment, relative to the module's base address as shown in the memory map.

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

1.4.7.2 JTAG TAP Controller (see page 38)

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

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

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

1.4.7.3 System Control and Clocks (see page 48)

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

1.4.8 Hardware Details

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

- "Pin Diagram" on page 339
- "Signal Tables" on page 340
- "Operating Characteristics" on page 347
- "Electrical Characteristics" on page 348
- "Package Information" on page 358

2 ARM Cortex-M3 Processor Core

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

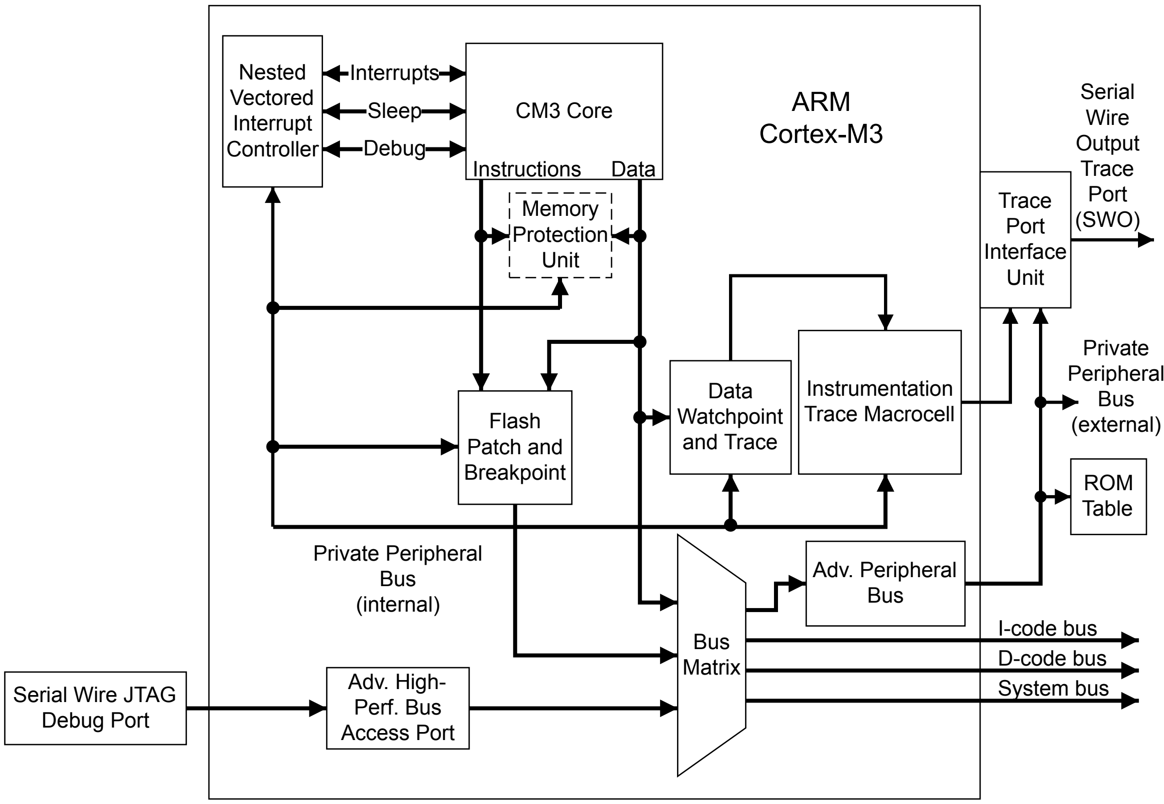
- Compact core.
- Thumb-2 instruction set, delivering the high-performance expected of an ARM core in the memory size usually associated with 8- and 16-bit devices; typically in the range of a few kilobytes of memory for microcontroller class applications.
- Rapid application execution through Harvard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

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

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

2.1 Block Diagram

Figure 2-1. CPU Block Diagram



2.2 Functional Description

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

Luminary Micro has implemented the ARM Cortex-M3 core as shown in Figure 2-1 on page 29. As noted in the *ARM® Cortex™-M3 Technical Reference Manual*, several Cortex-M3 components are flexible in their implementation: SW/JTAG-DP, ETM, TPIU, the ROM table, the MPU, and the Nested Vectored Interrupt Controller (NVIC). Each of these is addressed in the sections that follow.

2.2.1 Serial Wire and JTAG Debug

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

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

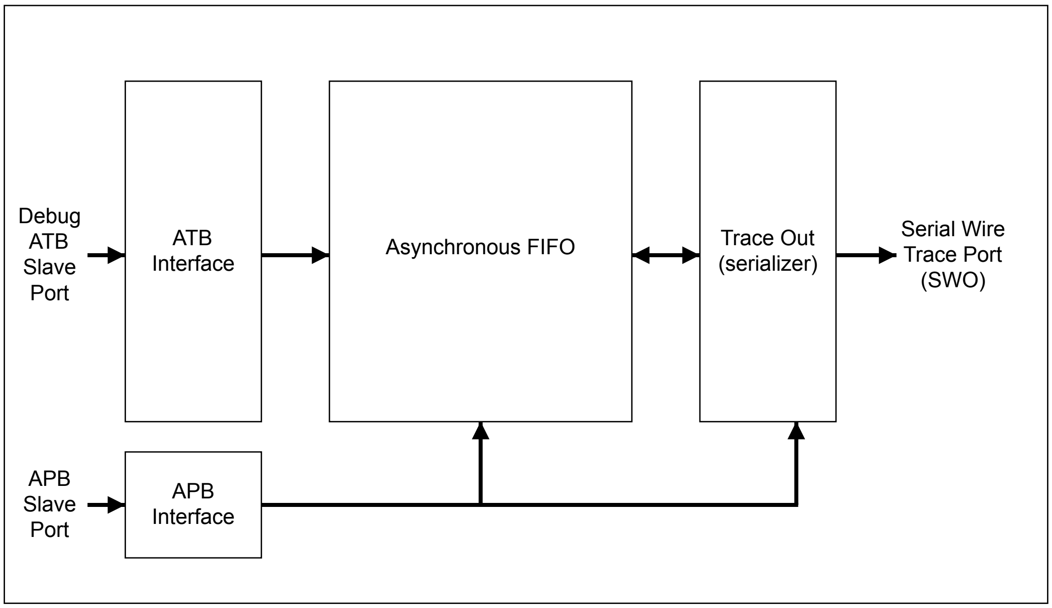
2.2.2 Embedded Trace Macrocell (ETM)

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

2.2.3 Trace Port Interface Unit (TPIU)

The TPIU acts as a bridge between the Cortex-M3 trace data from the ITM, and an off-chip Trace Port Analyzer. The Stellaris® devices have implemented TPIU as shown in Figure 2-2 on page 30. This is similar to the non-ETM version described in the *ARM® Cortex™-M3 Technical Reference Manual*, however, SWJ-DP only provides SWV output for the TPIU.

Figure 2-2. TPIU Block Diagram



2.2.4 ROM Table

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

2.2.5 Memory Protection Unit (MPU)

The Memory Protection Unit (MPU) is included on the LM3S600 controller and supports the standard ARMv7 Protected Memory System Architecture (PMSA) model. The MPU provides full support for protection regions, overlapping protection regions, access permissions, and exporting memory attributes to the system.

2.2.6 Nested Vectored Interrupt Controller (NVIC)

The Nested Vectored Interrupt Controller (NVIC):

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

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

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

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

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

2.2.6.1 Interrupts

The *ARM® Cortex™-M3 Technical Reference Manual* describes the maximum number of interrupts and interrupt priorities. The LM3S600 microcontroller supports 21 interrupts with eight priority levels.

2.2.6.2 System Timer (SysTick)

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

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

Functional Description

The timer consists of three registers:

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

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

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

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

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

SysTick Control and Status Register

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

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

SysTick Reload Value Register

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

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

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

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

SysTick Current Value Register

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

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

SysTick Calibration Value Register

The SysTick Calibration Value register is not implemented.

3 Memory Map

The memory map for the LM3S600 controller is provided in Table 3-1 on page 34.

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

Important: In Table 3-1 on page 34, addresses not listed are reserved.

Table 3-1. Memory Map[a]

Start	End	Description	For details on registers, see page ...
Memory			
0x0000.0000	0x0000.7FFF	On-chip flash [b]	108
0x2000.0000	0x2000.1FFF	Bit-banded on-chip SRAM[c]	108
0x2010.0000	0x200F.FFFF	Reserved	-
0x2200.0000	0x22003.FFFF	Bit-band alias of 0x2000.0000 through 0x200F.FFFF	103
0x2204.0000	0x23FF.FFFF	Reserved	-
FiRM Peripherals			
0x4000.0000	0x4000.0FFF	Watchdog timer	195
0x4000.4000	0x4000.4FFF	GPIO Port A	125
0x4000.5000	0x4000.5FFF	GPIO Port B	125
0x4000.6000	0x4000.6FFF	GPIO Port C	125
0x4000.7000	0x4000.7FFF	GPIO Port D	125
0x4000.8000	0x4000.8FFF	SSI0	265
0x4000.C000	0x4000.CFFF	UART0	222
0x4000.D000	0x4000.DFFF	UART1	222
Peripherals			
0x4002.0000	0x4002.07FF	I2C Master 0	304
0x4002.0800	0x4002.0FFF	I2C Slave 0	317
0x4002.4000	0x4002.7FFF	GPIO Port E	125
0x4003.0000	0x4003.0FFF	Timer0	168
0x4003.1000	0x4003.1FFF	Timer1	168
0x4003.2000	0x4003.2FFF	Timer2	168
0x4003.C000	0x4003.CFFF	Analog Comparators	326
0x400F.D000	0x400F.DFFF	Flash control	108
0x400F.E000	0x400F.FFFF	System control	56
0x4200.0000	0x43FF.FFFF	Bit-banded alias of 0x4000.0000 through 0x400F.FFFF	-
Private Peripheral Bus			

Start	End	Description	For details on registers, see page ...
0xE000.0000	0xE000.0FFF	Instrumentation Trace Macrocell (ITM)	*ARM® Cortex™-M3 Technical Reference Manual*
0xE000.1000	0xE000.1FFF	Data Watchpoint and Trace (DWT)	
0xE000.2000	0xE000.2FFF	Flash Patch and Breakpoint (FPB)	
0xE000.3000	0xE000.DFFF	Reserved	
0xE000.E000	0xE000.EFFF	Nested Vectored Interrupt Controller (NVIC)	
0xE000.F000	0xE003.FFFF	Reserved	
0xE004.0000	0xE004.0FFF	Trace Port Interface Unit (TPIU)	
0xE004.1000	0xE004.1FFF	Reserved	-
0xE004.2000	0xE00F.FFFF	Reserved	-
0xE010.0000	0xFFFF.FFFF	Reserved for vendor peripherals	-

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

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

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

4 Interrupts

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

Table 4-1 on page 36 lists all the exceptions. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 21 interrupts (listed in Table 4-2 on page 37).

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

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

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

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

Note: In Table 4-2 on page 37 interrupts not listed are reserved.

Table 4-1. Exception Types

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

Exception Type	Position	Priority[a]	Description
Debug Monitor	12	settable	Debug monitor (when not halting). This is synchronous, but only active when enabled. It does not activate if lower priority than the current activation.
-	13	-	Reserved.
PendSV	14	settable	Pendable request for system service. This is asynchronous and only pended by software.
SysTick	15	settable	System tick timer has fired. This is asynchronous.
Interrupts	16 and above	settable	Asserted from outside the ARM Cortex-M3 core and fed through the NVIC (prioritized). These are all asynchronous. Table 4-2 on page 37 lists the interrupts on the LM3S600 controller.

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

Table 4-2. Interrupts

Interrupt (Bit in Interrupt Registers)	Description
0	GPIO Port A
1	GPIO Port B
2	GPIO Port C
3	GPIO Port D
4	GPIO Port E
5	UART0
6	UART1
7	SSI0
8	I2C0
18	Watchdog timer
19	Timer0 A
20	Timer0 B
21	Timer1 A
22	Timer1 B
23	Timer2 A
24	Timer2 B
25	Analog Comparator 0
26	Analog Comparator 1
27	Analog Comparator 2
28	System Control
29	Flash Control
30-31	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 the standard five pins: $\overline{\text{TRST}}$, TCK, TMS, TDI, and TDO. Data is transmitted serially into the controller on TDI and out of the controller on TDO. The interpretation of this data is dependent on the current state of the TAP controller. For detailed information on the operation of the JTAG port and TAP controller, please refer to the *IEEE Standard 1149.1-Test Access Port and Boundary-Scan Architecture*.

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

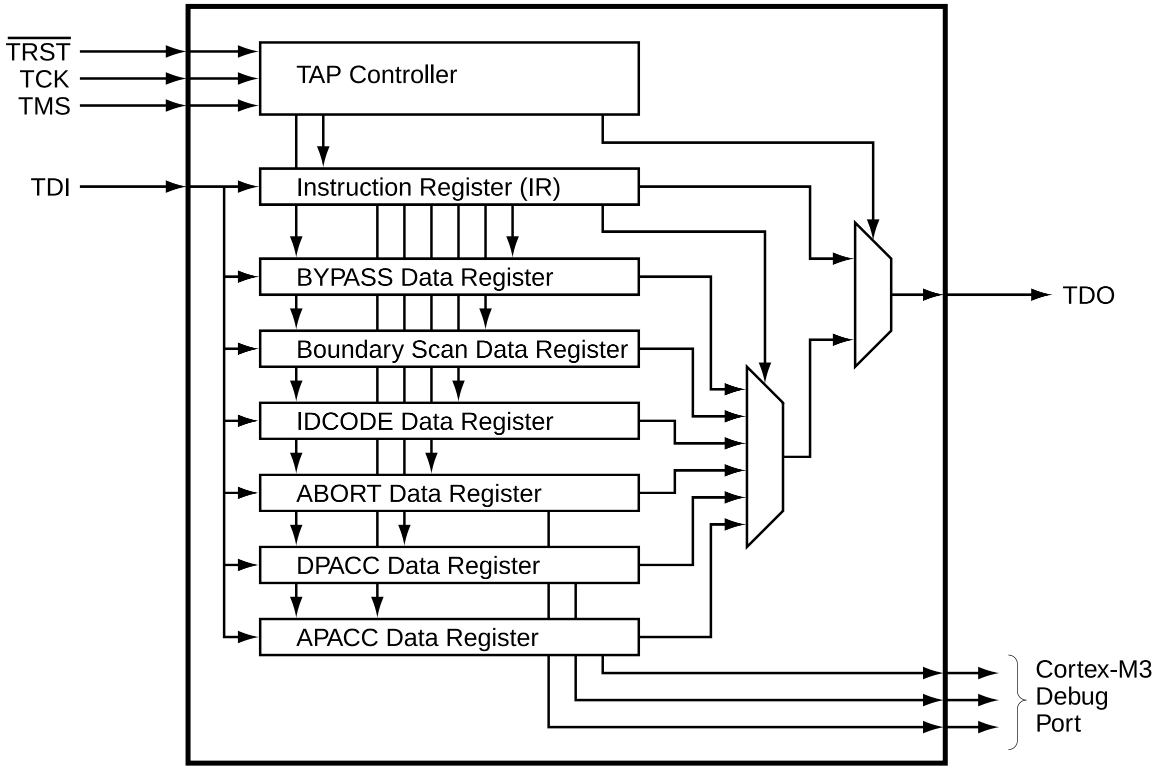
The JTAG module has the following features:

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

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

5.1 Block Diagram

Figure 5-1. JTAG Module Block Diagram



5.2 Functional Description

A high-level conceptual drawing of the JTAG module is shown in Figure 5-1 on page 39. The JTAG module is composed of the Test Access Port (TAP) controller and serial shift chains with parallel update registers. The TAP controller is a simple state machine controlled by the TRST, TCK and TMS inputs. The current state of the TAP controller depends on the current value of TRST and the sequence of values captured on TMS at the rising edge of TCK. The TAP controller determines when the serial shift chains capture new data, shift data from TDI towards TDO, and update the parallel load registers. The current state of the TAP controller also determines whether the Instruction Register (IR) chain or one of the Data Register (DR) chains is being accessed.

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

Some instructions, like EXTEST and INTEST, operate on data currently in a DR chain and do not capture, shift, or update any of the chains. Instructions that are not implemented decode to the BYPASS instruction to ensure that the serial path between TDI and TDO is always connected (see Table 5-2 on page 44 for a list of implemented instructions).

See “JTAG and Boundary Scan” on page 353 for JTAG timing diagrams.

5.2.1 JTAG Interface Pins

The JTAG interface consists of five standard pins: TRST, TCK, TMS, TDI, and TDO. These pins and their associated reset state are given in Table 5-1 on page 40. Detailed information on each pin follows.

Table 5-1. JTAG Port Pins Reset State

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

5.2.1.1 Test Reset Input (TRST)

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

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

5.2.1.2 Test Clock Input (TCK)

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

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

5.2.1.3 Test Mode Select (TMS)

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

Holding TMS high for five consecutive TCK cycles drives the TAP controller state machine to the Test-Logic-Reset state. When the TAP controller enters the Test-Logic-Reset state, the JTAG Instruction Register (IR) resets to the default instruction, IDCODE. Therefore, this sequence can be used as a reset mechanism, similar to asserting TRST. The JTAG Test Access Port state machine can be seen in its entirety in Figure 5-2 on page 42.

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

5.2.1.4 Test Data Input (TDI)

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

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

5.2.1.5 Test Data Output (TDO)

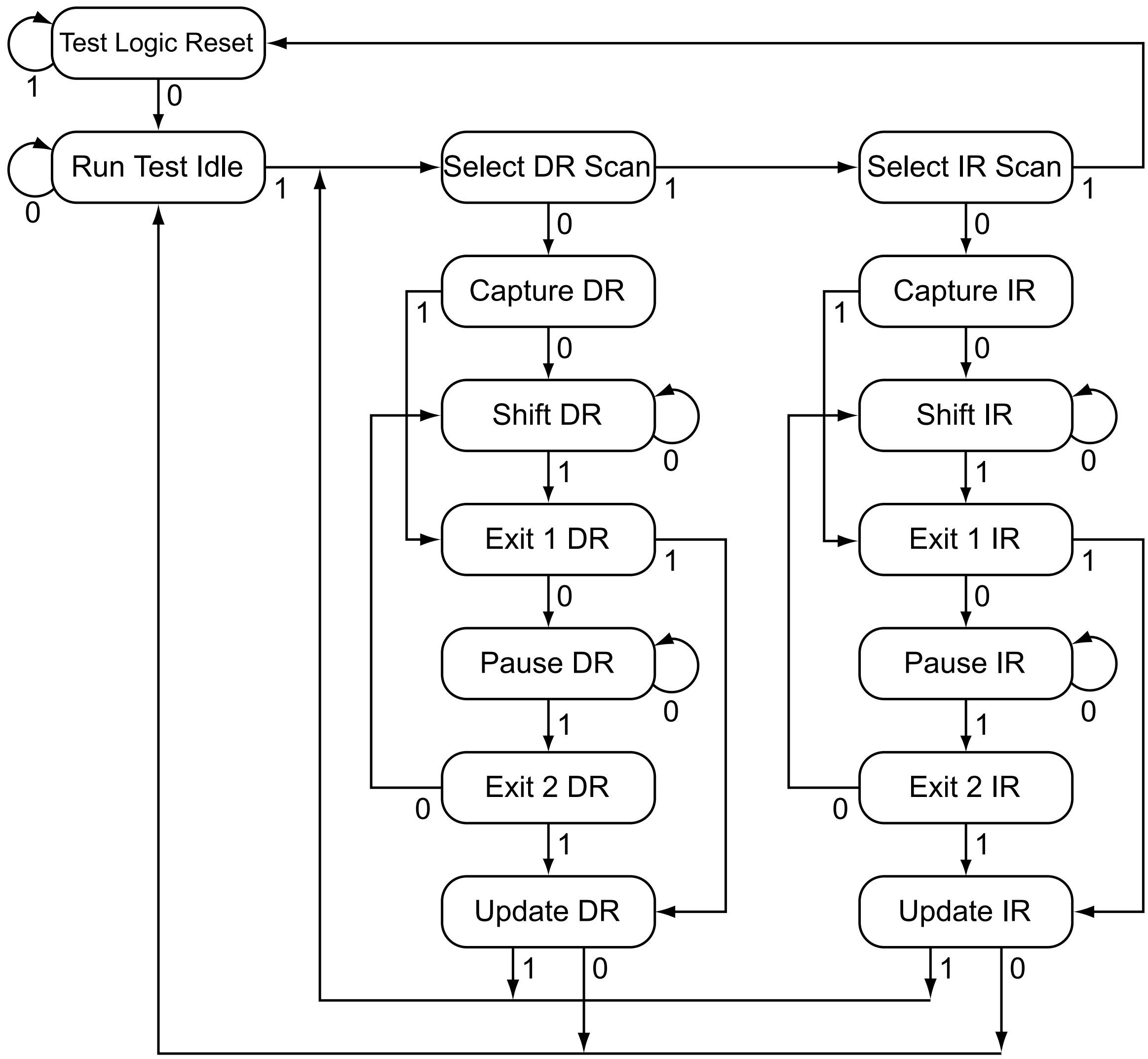
The TDO pin provides an output stream of serial information from the IR chain or the DR chains. The value of TDO depends on the current TAP state, the current instruction, and the data in the chain being accessed. In order to save power when the JTAG port is not being used, the TDO pin is placed in an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

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

5.2.2 JTAG TAP Controller

The JTAG TAP controller state machine is shown in Figure 5-2 on page 42. The TAP controller state machine is reset to the Test-Logic-Reset state on the assertion of a Power-On-Reset (POR) or the assertion of $\overline{\text{TRST}}$. Asserting the correct sequence on the TMS pin allows the JTAG module to shift in new instructions, shift in data, or idle during extended testing sequences. For detailed information on the function of the TAP controller and the operations that occur in each state, please refer to *IEEE Standard 1149.1*.

Figure 5-2. Test Access Port State Machine



5.2.3 Shift Registers

The Shift Registers consist of a serial shift register chain and a parallel load register. The serial shift register chain samples specific information during the TAP controller's CAPTURE states and allows this information to be shifted out of TDO during the TAP controller's SHIFT states. While the sampled data is being shifted out of the chain on TDO, new data is being shifted into the serial shift register on TDI. This new data is stored in the parallel load register during the TAP controller's UPDATE states. Each of the shift registers is discussed in detail in "Register Descriptions" on page 44.

5.2.4 Operational Considerations

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

5.2.4.1 GPIO Functionality

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

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

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

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

5.2.4.2 ARM Serial Wire Debug (SWD)

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

The preamble used to enable the SWD interface of the SWJ-DP module starts with the TAP controller in the Test-Logic-Reset state. From here, the preamble sequences the TAP controller through the following states: Run Test Idle, Select DR, Select IR, Capture IR, Exit1 IR, Update IR, Run Test Idle, Select DR, Select IR, Capture IR, Exit1 IR, Update IR, Run Test Idle, Select DR, Select IR, and Test-Logic-Reset states.

Stepping through the JTAG TAP Instruction Register (IR) load sequences of the TAP state machine twice without shifting in a new instruction enables the SWD interface and disables the JTAG interface. For more information on this operation and the SWD interface, see the *ARM® Cortex™-M3 Technical Reference Manual* and the *ARM® CoreSight Technical Reference Manual*.

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

5.3 Initialization and Configuration

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

5.4 Register Descriptions

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

5.4.1 Instruction Register (IR)

The JTAG TAP Instruction Register (IR) is a four-bit serial scan chain with a parallel load register connected between the JTAG `TDI` and `TDO` pins. When the TAP Controller is placed in the correct states, bits can be shifted into the Instruction Register. Once these bits have been shifted into the chain and updated, they are interpreted as the current instruction. The decode of the Instruction Register bits is shown in Table 5-2 on page 44. A detailed explanation of each instruction, along with its associated Data Register, follows.

Table 5-2. JTAG Instruction Register Commands

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

5.4.1.1 EXTEST Instruction

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

5.4.1.2 INTEST Instruction

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

5.4.1.3 SAMPLE/PRELOAD Instruction

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

While these samples of the inputs, outputs, and output enables are being shifted out of the Boundary Scan Data Register, new data is being shifted into the Boundary Scan Data Register from TDI. Once the new data has been shifted into the Boundary Scan Data Register, the data is saved in the parallel load registers when the TAP controller enters the Update DR state. This update of the parallel load register preloads data into the Boundary Scan Data Register that is associated with each input, output, and output enable. This preloaded data can be used with the EXTEST and INTEST instructions to drive data into or out of the controller. Please see “Boundary Scan Data Register” on page 46 for more information.

5.4.1.4 ABORT Instruction

The ABORT instruction connects the associated ABORT Data Register chain between TDI and TDO. This instruction provides read and write access to the ABORT Register of the ARM Debug Access Port (DAP). Shifting the proper data into this Data Register clears various error bits or initiates a DAP abort of a previous request. Please see the “ABORT Data Register” on page 47 for more information.

5.4.1.5 DPACC Instruction

The DPACC instruction connects the associated DPACC Data Register chain between TDI and TDO. This instruction provides read and write access to the DPACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to the ARM debug and status registers. Please see “DPACC Data Register” on page 47 for more information.

5.4.1.6 APACC Instruction

The APACC instruction connects the associated APACC Data Register chain between TDI and TDO. This instruction provides read and write access to the APACC Register of the ARM Debug Access Port (DAP). Shifting the proper data into this register and reading the data output from this register allows read and write access to internal components and buses through the Debug Port. Please see “APACC Data Register” on page 47 for more information.

5.4.1.7 IDCODE Instruction

The IDCODE instruction connects the associated IDCODE Data Register chain between TDI and TDO. This instruction provides information on the manufacturer, part number, and version of the ARM core. This information can be used by testing equipment and debuggers to automatically configure their input and output data streams. IDCODE is the default instruction that is loaded into the JTAG Instruction Register when a power-on-reset (POR) is asserted, $\overline{\text{TRST}}$ is asserted, or the Test-Logic-Reset state is entered. Please see “IDCODE Data Register” on page 46 for more information.

5.4.1.8 BYPASS Instruction

The BYPASS instruction connects the associated BYPASS Data Register chain between TDI and TDO. This instruction is used to create a minimum length serial path between the TDI and TDO ports. The BYPASS Data Register is a single-bit shift register. This instruction improves test efficiency by allowing components that are not needed for a specific test to be bypassed in the JTAG scan chain by loading them with the BYPASS instruction. Please see “BYPASS Data Register” on page 46 for more information.

5.4.2 Data Registers

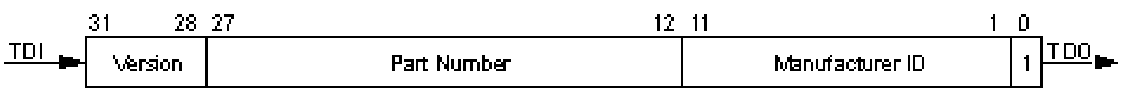
The JTAG module contains six Data Registers. These include: IDCODE, BYPASS, Boundary Scan, APACC, DPACC, and ABORT serial Data Register chains. Each of these Data Registers is discussed in the following sections.

5.4.2.1 IDCODE Data Register

The format for the 32-bit IDCODE Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 5-3 on page 46. The standard requires that every JTAG-compliant device implement either the IDCODE instruction or the BYPASS instruction as the default instruction. The LSB of the IDCODE Data Register is defined to be a 1 to distinguish it from the BYPASS instruction, which has an LSB of 0. This allows auto configuration test tools to determine which instruction is the default instruction.

The major uses of the JTAG port are for manufacturer testing of component assembly, and program development and debug. To facilitate the use of auto-configuration debug tools, the IDCODE instruction outputs a value of 0x1BA00477. This value indicates an ARM Cortex-M3, Version 1 processor. This allows the debuggers to automatically configure themselves to work correctly with the Cortex-M3 during debug.

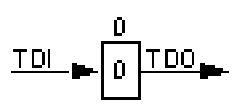
Figure 5-3. IDCODE Register Format



5.4.2.2 BYPASS Data Register

The format for the 1-bit BYPASS Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 5-4 on page 46. The standard requires that every JTAG-compliant device implement either the BYPASS instruction or the IDCODE instruction as the default instruction. The LSB of the BYPASS Data Register is defined to be a 0 to distinguish it from the IDCODE instruction, which has an LSB of 1. This allows auto configuration test tools to determine which instruction is the default instruction.

Figure 5-4. BYPASS Register Format

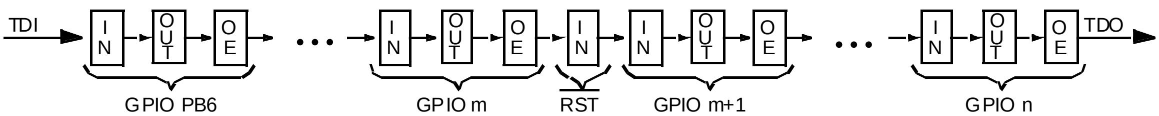


5.4.2.3 Boundary Scan Data Register

The format of the Boundary Scan Data Register is shown in Figure 5-5 on page 47. Each GPIO pin, in a counter-clockwise direction from the JTAG port pins, is included in the Boundary Scan Data Register. Each GPIO pin has three associated digital signals that are included in the chain. These signals are input, output, and output enable, and are arranged in that order as can be seen in the figure. In addition to the GPIO pins, the controller reset pin, $\overline{RST}$, is included in the chain. Because the reset pin is always an input, only the input signal is included in the Data Register chain.

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

Figure 5-5. Boundary Scan Register Format



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

5.4.2.4 APACC Data Register

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

5.4.2.5 DPACC Data Register

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

5.4.2.6 ABORT Data Register

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

6 System Control

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

6.1 Functional Description

The System Control module provides the following capabilities:

- Device identification, see "Device Identification" on page 48
- Local control, such as reset (see "Reset Control" on page 48), power (see "Power Control" on page 51) and clock control (see "Clock Control" on page 51)
- System control (Run, Sleep, and Deep-Sleep modes), see "System Control" on page 54

6.1.1 Device Identification

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

6.1.2 Reset Control

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

6.1.2.1 Reset Sources

The controller has six sources of reset:

1. External reset input pin ($\overline{\text{RST}}$) assertion, see "$\overline{\text{RST}}$ Pin Assertion" on page 48.
2. Power-on reset (POR), see "Power-On Reset (POR)" on page 49.
3. Internal brown-out (BOR) detector, see "Brown-Out Reset (BOR)" on page 49.
4. Software-initiated reset (with the software reset registers), see "Software Reset" on page 50.
5. A watchdog timer reset condition violation, see "Watchdog Timer Reset" on page 51.
6. Internal low drop-out (LDO) regulator output

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 external reset is the cause, and then all the other bits in the **RESC** register are cleared.

Note: The main oscillator is used for external resets and power-on resets; the internal oscillator is used during the internal process by internal reset and clock verification circuitry.

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

The external reset pin ($\overline{\text{RST}}$) resets the controller. This resets the core and all the peripherals except the JTAG TAP controller (see "JTAG Interface" on page 38). The external reset sequence is as follows:

1. The external reset pin ($\overline{RST}$) is asserted and then de-asserted.
2. After $\overline{RST}$ is de-asserted, the main crystal oscillator is allowed to settle and there is an internal main oscillator counter that takes from 15-30 ms to account for this. During this time, internal reset to the rest of the controller is held active.
3. The internal reset is released and the core fetches and loads the initial stack pointer, the initial program counter, the first instruction designated by the program counter, and begins execution.

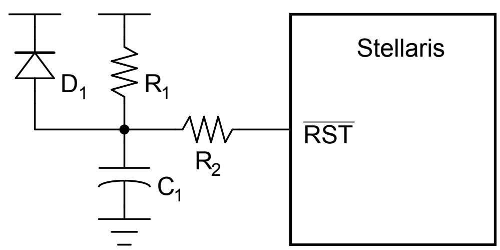
The external reset timing is shown in Figure 18-9 on page 356.

6.1.2.3 Power-On Reset (POR)

The Power-On Reset (POR) circuitry detects a rise in power-supply voltage (V_{DD}) and generates an on-chip reset pulse. To use the on-chip circuitry, the $\overline{RST}$ input needs to be connected to the power supply (V_{DD}) through a pull-up resistor (1K to 10K Ω).

The device must be operating within the specified operating parameters at the point when the on-chip power-on reset pulse is complete. The specified operating parameters include supply voltage, frequency, temperature, and so on. If the operating conditions are not met at the point of POR end, the Stellaris® controller does not operate correctly. In this case, the reset must be extended using external circuitry. The $\overline{RST}$ input may be used with the circuit as shown in Figure 6-1 on page 49.

Figure 6-1. External Circuitry to Extend Reset



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

The Power-On Reset sequence is as follows:

1. The controller waits for the later of external reset ($\overline{RST}$) or internal POR to go inactive.
2. After the resets are inactive, the main crystal oscillator is allowed to settle and there is an internal main oscillator counter that takes from 15-30 ms to account for this. During this time, internal reset to the rest of the controller is held active.
3. The internal reset is released and the core fetches and loads the initial stack pointer, the initial program counter, the first instruction designated by the program counter, and begins execution.

The internal POR is only active on the initial power-up of the controller. The Power-On Reset timing is shown in Figure 18-10 on page 356.

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

6.1.2.4 Brown-Out Reset (BOR)

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

The system provides a brown-out detection circuit that triggers if the power supply (V_{DD}) drops below a brown-out threshold voltage (V_{BTH}). The circuit is provided to guard against improper operation of logic and peripherals that operate off the power supply voltage (V_{DD}) and not the LDO voltage. If a brown-out condition is detected, the system may generate a controller interrupt or a system reset. The BOR circuit has a digital filter that protects against noise-related detection for the interrupt condition. This feature may be optionally enabled.

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

The brown-out reset sequence is as follows:

1. When V_{DD} drops below V_{BTH}, an internal BOR condition is set.
2. If the `BORWT` bit in the **PBORCTL** register is set and `BORIOR` is not set, the BOR condition is resampled again, after a delay specified by `BORTIM`, to determine if the original condition was caused by noise. If the BOR condition is not met the second time, then no further action is taken.
3. If the BOR condition exists, an internal reset is asserted.
4. The internal reset is released and the controller fetches and loads the initial stack pointer, the initial program counter, the first instruction designated by the program counter, and begins execution.
5. The internal BOR condition is reset after 500 µs to prevent another BOR condition from being set before software has a chance to investigate the original cause.

The internal Brown-Out Reset timing is shown in Figure 18-11 on page 357.

6.1.2.5 Software Reset

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

Peripherals can be individually reset by software via three registers that control reset signals to each peripheral (see the **SRCRn** registers). If the bit position corresponding to a peripheral is set and subsequently cleared, the peripheral is reset. The encoding of the reset registers is consistent with the encoding of the clock gating control for peripherals and on-chip functions (see “System Control” on page 54). Note that all reset signals for all clocks of the specified unit are asserted as a result of a software-initiated reset.

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

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

The software-initiated system reset timing is shown in Figure 18-12 on page 357.

6.1.2.6 Watchdog Timer Reset

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

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

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

The watchdog reset timing is shown in Figure 18-13 on page 357.

6.1.2.7 Low Drop-Out

A reset can be initiated when the internal low drop-out (LDO) regulator output goes unregulated. This is initially disabled and may be enabled by software. LDO is controlled with the **LDO Power Control (LDOPCTL)** register. The LDO reset sequence is as follows:

1. LDO goes unregulated and the LDOARST bit in the **LDOARST** register is set.
2. An internal reset is asserted.
3. The internal reset is released and the controller fetches and loads the initial stack pointer, the initial program counter, the first instruction designated by the program counter, and begins execution.

The LDO reset timing is shown in Figure 18-14 on page 357.

6.1.3 Power Control

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

6.1.4 Clock Control

System control determines the control of clocks in this part.

6.1.4.1 Fundamental Clock Sources

There are two clock sources for use in the device:

- **Internal Oscillator (IOSC):** The internal oscillator is an on-chip clock source. It does not require the use of any external components. The frequency of the internal oscillator is 12 MHz ± 30%.

Applications that do not depend on accurate clock sources may use this clock source to reduce system cost.

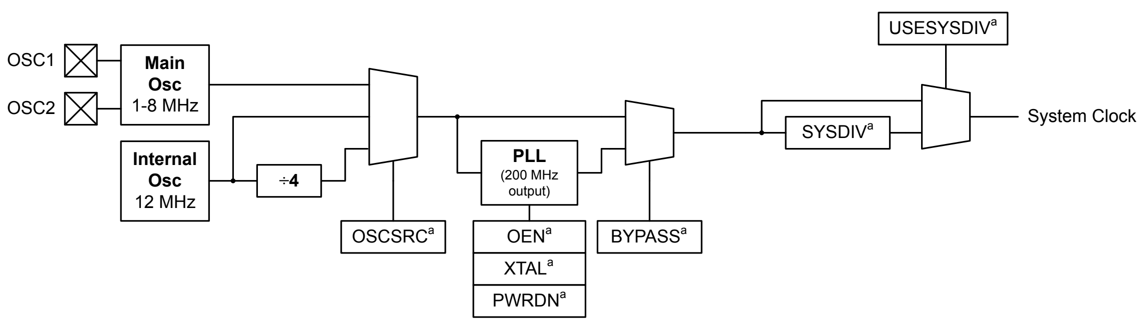
- **Main Oscillator:** The main oscillator provides a frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the OSC0 input pin, or an external crystal is connected across the OSC0 input and OSC1 output pins. The crystal value allowed depends on whether the main oscillator is used as the clock reference source to the PLL. If so, the crystal must be one of the supported frequencies between 3.579545 MHz through 8.192 MHz (inclusive). If the PLL is not being used, the crystal may be any one of the supported frequencies between 1 MHz and 8.192 MHz. The single-ended clock source range is from DC through the specified speed of the device. The supported crystals are listed in the XTAL bit in the **RCC** register (see page 66).

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

Nearly all of the control for the clocks is provided by the **Run-Mode Clock Configuration (RCC)** register.

Figure 6-2 on page 52 shows the logic for the main clock tree. The peripheral blocks are driven by the system clock signal and can be programmatically enabled/disabled.

Figure 6-2. Main Clock Tree



a. These are bit fields within the **Run-Mode Clock Configuration (RCC)** register.

6.1.4.2 Crystal Configuration for the Main Oscillator (MOSC)

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

The XTAL bit in the **RCC** register (see page 66) describes the available crystal choices and default programming values.

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

6.1.4.3 PLL Frequency Configuration

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

If the main oscillator provides the clock reference to the PLL, the translation provided by hardware and used to program the PLL is available for software in the **XTAL to PLL Translation (PLLCFG)** register (see page 70). The internal translation provides a translation within ± 1% of the targeted PLL VCO frequency.

The XTAL bit in the **RCC** register (see page 66) describes the available crystal choices and default programming of the **PLLCFG** register. The crystal number is written into the XTAL field of the **Run-Mode Clock Configuration (RCC)** register. Any time the XTAL field changes, the new settings are translated and the internal PLL settings are updated.

6.1.4.4 PLL Modes

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

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

The modes are programmed using the **RCC** register fields (see page 66).

6.1.4.5 PLL Operation

If the PLL configuration is changed, the PLL output frequency is unstable until it reconverges (relocks) to the new setting. The time between the configuration change and relock is T_{READY} (see Table 18-6 on page 350). During this time, the PLL is not usable as a clock reference.

The PLL is changed by one of the following:

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

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

6.1.4.6 Clock Verification Timers

There are three identical clock verification circuits that can be enabled though software. The circuit checks the faster clock by a slower clock using timers:

- The main oscillator checks the PLL.
- The main oscillator checks the internal oscillator.
- The internal oscillator divided by 64 checks the main oscillator.

If the verification timer function is enabled and a failure is detected, the main clock tree is immediately switched to a working clock and an interrupt is generated to the controller. Software can then determine the course of action to take. The actual failure indication and clock switching does not clear without a write to the **CLKVCLR** register, an external reset, or a POR reset. The clock verification timers are controlled by the `PLLVER` , `IOSCVER` , and `MOSCVER` bits in the **RCC** register.

6.1.5 System Control

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

In Run mode, the controller is actively executing code. In Sleep mode, the clocking of the device is unchanged but the controller no longer executes code (and is no longer clocked). In Deep-Sleep mode, the clocking of the device may change (depending on the Run mode clock configuration) and the controller no longer executes code (and is no longer clocked). An interrupt returns the device to Run mode from one of the sleep modes. Each mode is described in more detail in this section.

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

- **Run Mode.** Run mode provides normal operation of the processor and all of the peripherals that are currently enabled by the **RCGCn** registers. The system clock can be any of the available clock sources including the PLL.

- **Sleep Mode.** Sleep mode is entered by the Cortex-M3 core executing a `WFI (Wait for Interrupt)` instruction. Any properly configured interrupt event in the system will bring the processor back into Run mode. See the system control NVIC section of the *ARM® Cortex™-M3 Technical Reference Manual* for more details.

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

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

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

6.2 Initialization and Configuration

The PLL is configured using direct register writes to the **RCC** register. The steps required to successfully change the PLL-based system clock are:

1. Bypass the PLL and system clock divider by setting the BYPASS bit and clearing the USESYS bit in the **RCC** register. This configures the system to run off a “raw” clock source (using the main oscillator or internal oscillator) and allows for the new PLL configuration to be validated before switching the system clock to the PLL.
2. Select the crystal value (XTAL) and oscillator source (OSCSRC), and clear the PWRDN and OEN bits in **RCC**. Setting the XTAL field automatically pulls valid PLL configuration data for the appropriate crystal, and clearing the PWRDN and OEN bits powers and enables the PLL and its output.
3. Select the desired system divider (SYSDIV) in **RCC** 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**.

Note: If the BYPASS bit is cleared before the PLL locks, it is possible to render the device unusable.

6.3 Register Map

Table 6-1 on page 55 lists the System Control registers, grouped by function. The offset listed is a hexadecimal increment to the register’s address, relative to the System Control base address of 0x400F.E000.

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

Table 6-1. System Control Register Map

Offset	Name	Type	Reset	Description	See page
0x000	DID0	RO	-	Device Identification 0	57
0x004	DID1	RO	-	Device Identification 1	74
0x008	DC0	RO	0x001F.000F	Device Capabilities 0	76
0x010	DC1	RO	0x0000.309F	Device Capabilities 1	77
0x014	DC2	RO	0x0707.1013	Device Capabilities 2	79
0x018	DC3	RO	0x3F00.7FC0	Device Capabilities 3	81
0x01C	DC4	RO	0x0000.001F	Device Capabilities 4	83
0x030	PBORCTL	R/W	0x0000.7FFD	Power-On and Brown-Out Reset Control	59
0x034	LDOPCTL	R/W	0x0000.0000	LDO Power Control	60
0x040	SRCR0	R/W	0x00000000	Software Reset Control 0	99
0x044	SRCR1	R/W	0x00000000	Software Reset Control 1	100
0x048	SRCR2	R/W	0x00000000	Software Reset Control 2	102
0x050	RIS	RO	0x0000.0000	Raw Interrupt Status	61
0x054	IMC	R/W	0x0000.0000	Interrupt Mask Control	62

Offset	Name	Type	Reset	Description	See page
0x058	MISC	R/W1C	0x0000.0000	Masked Interrupt Status and Clear	64
0x05C	RESC	R/W	-	Reset Cause	65
0x060	RCC	R/W	0x07A0.3AD1	Run-Mode Clock Configuration	66
0x064	PLLCFG	RO	-	XTAL to PLL Translation	70
0x100	RCGC0	R/W	0x00000040	Run Mode Clock Gating Control Register 0	84
0x104	RCGC1	R/W	0x00000000	Run Mode Clock Gating Control Register 1	87
0x108	RCGC2	R/W	0x00000000	Run Mode Clock Gating Control Register 2	93
0x110	SCGC0	R/W	0x00000040	Sleep Mode Clock Gating Control Register 0	85
0x114	SCGC1	R/W	0x00000000	Sleep Mode Clock Gating Control Register 1	89
0x118	SCGC2	R/W	0x00000000	Sleep Mode Clock Gating Control Register 2	95
0x120	DCGC0	R/W	0x00000040	Deep Sleep Mode Clock Gating Control Register 0	86
0x124	DCGC1	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 1	91
0x128	DCGC2	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 2	97
0x144	DSLPCLKCFG	R/W	0x0780.0000	Deep Sleep Clock Configuration	71
0x150	CLKVCLR	R/W	0x0000.0000	Clock Verification Clear	72
0x160	LDOARST	R/W	0x0000.0000	Allow Unregulated LDO to Reset the Part	73

6.4 Register Descriptions

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

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

This register identifies the version of the device.

Device Identification 0 (DID0)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved	VER			reserved											
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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	0x0	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: Value Description 0x0 Initial **DID0** register format definition for Stellaris® Sandstorm-class devices.
27:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	MAJOR	RO	-	Major Revision This field specifies the major revision number of the device. The major revision reflects changes to base layers of the design. The major revision number is indicated in the part number as a letter (A for first revision, B for second, and so on). This field is encoded as follows: Value Description 0x0 Revision A (initial device) 0x1 Revision B (first base layer revision) 0x2 Revision C (second base layer revision) and so on.

Bit/Field	Name	Type	Reset	Description
7:0	MINOR	RO	-	Minor Revision This field specifies the minor revision number of the device. The minor revision reflects changes to the metal layers of the design. The `MINOR` field value is reset when the `MAJOR` field is changed. This field is numeric and is encoded as follows: 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: Power-On and Brown-Out Reset Control (PBORCTL), offset 0x030

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

Power-On and 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
	BORTIM														BORIOR	BORWT
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	0	1

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:2	BORTIM	R/W	0x1FFF	BOR Time Delay This field specifies the number of internal oscillator clocks delayed before the BOR output is resampled if the `BORWT` bit is set. The width of this field is derived by the t_{BOR} width of 500 µs and the internal oscillator (IOSC) frequency of 12 MHz ± 30%. At +30%, the counter value has to exceed 7,800.
1	BORIOR	R/W	0	BOR Interrupt or Reset This bit controls how a BOR event is signaled to the controller. If set, a reset is signaled. Otherwise, an interrupt is signaled.
0	BORWT	R/W	1	BOR Wait and Check for Noise This bit specifies the response to a brown-out signal assertion if `BORIOR` is not set. If `BORWT` is set to 1 and `BORIOR` is cleared to 0, the controller waits `BORTIM` IOSC periods and resamples the BOR output. If still asserted, a BOR interrupt is signalled. If no longer asserted, the initial assertion is suppressed (attributable to noise). If `BORWT` is 0, BOR assertions do not resample the output and any condition is reported immediately if enabled.

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

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

LDO Power Control (LDOPCTL)

Base 0x400F.E000
Offset 0x034
Type R/W, reset 0x0000.0000

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

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

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

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

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

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

Raw Interrupt Status (RIS)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									PLLLRIS	CLRIS	IOFRIS	MOFRIS	LDORIS	BORRIS	PLLFRIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PLLLRIS	RO	0	PLL Lock Raw Interrupt Status This bit is set when the PLL T_{READY} Timer asserts.
5	CLRIS	RO	0	Current Limit Raw Interrupt Status This bit is set if the LDO's CLE output asserts.
4	IOFRIS	RO	0	Internal Oscillator Fault Raw Interrupt Status This bit is set if an internal oscillator fault is detected.
3	MOFRIS	RO	0	Main Oscillator Fault Raw Interrupt Status This bit is set if a main oscillator fault is detected.
2	LDORIS	RO	0	LDO Power Unregulated Raw Interrupt Status This bit is set if a LDO voltage is unregulated.
1	BORRIS	RO	0	Brown-Out Reset Raw Interrupt Status This bit is the raw interrupt status for any brown-out conditions. If set, a brown-out condition is currently active. This is an unregistered signal from the brown-out detection circuit. An interrupt is reported if the `BORIM` bit in the **IMC** register is set and the `BORIOR` bit in the **PBORCTL** register is cleared.
0	PLLFRIS	RO	0	PLL Fault Raw Interrupt Status This bit is set if a PLL fault is detected (stops oscillating).

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

Central location for system control interrupt masks.

Interrupt Mask Control (IMC)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									PLLLIM	CLIM	IOFIM	MOFIM	LDOIM	BORIM	PLLFIM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PLLLIM	R/W	0	PLL Lock Interrupt Mask This bit specifies whether a current limit detection is promoted to a controller interrupt. If set, an interrupt is generated if `PLLLRIS` in **RIS** is set; otherwise, an interrupt is not generated.
5	CLIM	R/W	0	Current Limit Interrupt Mask This bit specifies whether a current limit detection is promoted to a controller interrupt. If set, an interrupt is generated if `CLRIS` is set; otherwise, an interrupt is not generated.
4	IOFIM	R/W	0	Internal Oscillator Fault Interrupt Mask This bit specifies whether an internal oscillator fault detection is promoted to a controller interrupt. If set, an interrupt is generated if `IOFRIS` is set; otherwise, an interrupt is not generated.
3	MOFIM	R/W	0	Main Oscillator Fault Interrupt Mask This bit specifies whether a main oscillator fault detection is promoted to a controller interrupt. If set, an interrupt is generated if `MOFRIS` is set; otherwise, an interrupt is not generated.
2	LDOIM	R/W	0	LDO Power Unregulated Interrupt Mask This bit specifies whether an LDO unregulated power situation is promoted to a controller interrupt. If set, an interrupt is generated if `LDORIS` is set; otherwise, an interrupt is not generated.
1	BORIM	R/W	0	Brown-Out Reset Interrupt Mask This bit specifies whether a brown-out condition is promoted to a controller interrupt. If set, an interrupt is generated if `BORRIS` is set; otherwise, an interrupt is not generated.

Bit/Field	Name	Type	Reset	Description
0	PLLFIM	R/W	0	PLL Fault Interrupt Mask This bit specifies whether a PLL fault detection is promoted to a controller interrupt. If set, an interrupt is generated if PLLFRIS is set; otherwise, an interrupt is not generated.

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

Central location for system control result of RIS AND IMC to generate an interrupt to the controller. All of the bits are R/W1C and this action also clears the corresponding raw interrupt bit in the **RIS** register (see page 61).

Masked Interrupt Status and Clear (MISC)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									PLLLMIS	CLMIS	IOFMIS	MOFMIS	LDOMIS	BORMIS	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	R/W1C	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	PLLLMIS	R/W1C	0	PLL Lock Masked Interrupt Status This bit is set when the PLL T_{READY} timer asserts. The interrupt is cleared by writing a 1 to this bit.
5	CLMIS	R/W1C	0	Current Limit Masked Interrupt Status This bit is set if the LDO's CLE output asserts. The interrupt is cleared by writing a 1 to this bit.
4	IOFMIS	R/W1C	0	Internal Oscillator Fault Masked Interrupt Status This bit is set if an internal oscillator fault is detected. The interrupt is cleared by writing a 1 to this bit.
3	MOFMIS	R/W1C	0	Main Oscillator Fault Masked Interrupt Status This bit is set if a main oscillator fault is detected. The interrupt is cleared by writing a 1 to this bit.
2	LDOMIS	R/W1C	0	LDO Power Unregulated Masked Interrupt Status This bit is set if LDO power is unregulated. The interrupt is cleared by writing a 1 to this bit.
1	BORMIS	R/W1C	0	BOR Masked Interrupt Status This bit is the masked interrupt status for any brown-out conditions. If set, a brown-out condition was detected. An interrupt is reported if the `BORIM` bit in the IMC register is set and the `BORIOR` bit in the **PBORCTL** register is cleared. The interrupt is cleared by writing a 1 to 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 7: Reset Cause (RESC), offset 0x05C

This field specifies the cause of the reset event to software. The reset value is determined by the cause of the reset. When an external reset is the cause (EXT is set), all other reset bits are cleared. However, if the reset is due to any other cause, the remaining bits are sticky, allowing software to see all causes.

Reset Cause (RESC)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										LDO	SW	WDT	BOR	POR	EXT
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	-	-	-	-	-	-

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

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

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

Run-Mode Clock Configuration (RCC)

Base 0x400F.E000
Offset 0x060
Type R/W, reset 0x07A0.3AD1

	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	OEN	BYPASS	PLLVER	XTAL				OSCSRC		IOSCVER	MOSCVER	IOSCDIS	MOSCDIS
Type	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	1	1	0	1	0	1	1	0	0	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 controller enters a Sleep or Deep-Sleep mode (respectively). If set, the **SCGCn** or **DCGCn** registers are used to control the clocks distributed to the peripherals when the controller is in a sleep mode. Otherwise, the **Run-Mode Clock Gating Control (RCGCn)** registers are used when the controller enters a sleep mode. The **RCGCn** registers are always used to control the clocks in Run mode. This allows peripherals to consume less power when the controller is in a sleep mode and the peripheral is unused.

Bit/Field	Name	Type	Reset	Description
26:23	SYSDIV	R/W	0xF	System Clock Divisor Specifies which divisor is used to generate the system clock from the PLL output. The PLL VCO frequency is 200 MHz. Value / Divisor (BYPASS=1) / Frequency (BYPASS=0): 0x0 reserved reserved 0x1 /2 reserved 0x2 /3 reserved 0x3 /4 50 MHz 0x4 /5 40 MHz 0x5 /6 33.33 MHz 0x6 /7 28.57 MHz 0x7 /8 25 MHz 0x8 /9 22.22 MHz 0x9 /10 20 MHz 0xA /11 18.18 MHz 0xB /12 16.67 MHz 0xC /13 15.38 MHz 0xD /14 14.29 MHz 0xE /15 13.33 MHz 0xF /16 12.5 MHz (default) When reading the **Run-Mode Clock Configuration (RCC)** register (see page 66), the `SYSDIV` value is `MINSYSDIV` if a lower divider was requested and the PLL is being used. This lower value is allowed to divide a non-PLL source.
22	USESYSDIV	R/W	0	Enable System Clock Divider Use the system clock divider as the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source.
21:14	reserved	RO	0	Software should not rely on the value of 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 This bit connects to the PLL PWRDN input. The reset value of 1 powers down the PLL. See Table 6-2 on page 69 for PLL mode control.
12	OEN	R/W	1	PLL Output Enable This bit specifies whether the PLL output driver is enabled. If cleared, the driver transmits the PLL clock to the output. Otherwise, the PLL clock does not oscillate outside the PLL module. **Note:** Both `PWRDN` and `OEN` must be cleared to run the PLL.

Bit/Field	Name	Type	Reset	Description
11	BYPASS	R/W	1	PLL Bypass Chooses whether the system clock is derived from the PLL output or the OSC source. If set, the clock that drives the system is the OSC source. Otherwise, the clock that drives the system is the PLL output clock divided by the system divider.
10	PLLVER	R/W	0	PLL Verification This bit controls the PLL verification timer function. If set, the verification timer is enabled and an interrupt is generated if the PLL becomes inoperative. Otherwise, the verification timer is not enabled.
9:6	XTAL	R/W	0xB	Crystal Value This field specifies the crystal value attached to the main oscillator. The encoding for this field is provided below.

Value	Crystal Frequency (MHz) Not Using the PLL	Crystal Frequency (MHz) Using the PLL
0x0	1.000	reserved
0x1	1.8432	reserved
0x2	2.000	reserved
0x3	2.4576	reserved
0x4	3.579545 MHz	
0x5	3.6864 MHz	
0x6	4 MHz	
0x7	4.096 MHz	
0x8	4.9152 MHz	
0x9	5 MHz	
0xA	5.12 MHz	
0xB	6 MHz (reset value)	
0xC	6.144 MHz	
0xD	7.3728 MHz	
0xE	8 MHz	
0xF	8.192 MHz	

Bit/Field	Name	Type	Reset	Description
5:4	OSCSRC	R/W	0x0	Oscillator Source Picks among the four input sources for the OSC. The values are:

Value	Input Source
0x0	Main oscillator (default)
0x1	Internal oscillator (default)
0x2	Internal oscillator / 4 (this is necessary if used as input to PLL)
0x3	reserved

Bit/Field	Name	Type	Reset	Description
3	IOSCVER	R/W	0	Internal Oscillator Verification Timer This bit controls the internal oscillator verification timer function. If set, the verification timer is enabled and an interrupt is generated if the timer becomes inoperative. Otherwise, the verification timer is not enabled.

Bit/Field	Name	Type	Reset	Description
2	MOSCVER	R/W	0	Main Oscillator Verification Timer This bit controls the main oscillator verification timer function. If set, the verification timer is enabled and an interrupt is generated if the timer becomes inoperative. Otherwise, the verification timer is not enabled.
1	IOSCDIS	R/W	0	Internal Oscillator Disable 0: Internal oscillator (IOSC) is enabled. 1: Internal oscillator is disabled.
0	MOSCDIS	R/W	1	Main Oscillator Disable 0: Main oscillator is enabled. 1: Main oscillator is disabled (default).

Table 6-2. PLL Mode Control

PWRDN	OEN	Mode
1	X	Power down
0	0	Normal

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

This register provides a means of translating external crystal frequencies into the appropriate PLL settings. This register is initialized during the reset sequence and updated anytime that the XTAL field changes in the **Run-Mode Clock Configuration (RCC)** register (see page 66).

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

```
PLLFreq = OSCFreq * (F + 2) / (R + 2)
```

XTAL to PLL Translation (PLLCFG)

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

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

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:14	OD	RO	-	PLL OD Value This field specifies the value supplied to the PLL's OD input. Value Description 0x0 Divide by 1 0x1 Divide by 2 0x2 Divide by 4 0x3 Reserved
13:5	F	RO	-	PLL F Value This field specifies the value supplied to the PLL's F input.
4:0	R	RO	-	PLL R Value This field specifies the value supplied to the PLL's R input.

Register 10: Deep Sleep Clock Configuration (DSLPCLKCFG), offset 0x144

This register is used to automatically switch from the main oscillator to the internal oscillator when entering Deep-Sleep mode. The system clock source is the main oscillator by default. When this register is set, the internal oscillator is powered up and the main oscillator is powered down. 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.

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															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IOSC	R/W	0	IOSC Clock Source When set, forces IOSC to be clock source during Deep-Sleep (overrides `DSOSCSRC` field if set)

Register 11: Clock Verification Clear (CLKVCLR), offset 0x150

This register is provided as a means of clearing the clock verification circuits by software. Since the clock verification circuits force a known good clock to control the process, the controller is allowed the opportunity to solve the problem and clear the verification fault. This register clears all clock verification faults. To clear a clock verification fault, the VERCLR bit must be set and then cleared by software. This bit is not self-clearing.

Clock Verification Clear (CLKVCLR)

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
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	VERCLR	R/W	0	Clock Verification Clear Clears clock verification faults.

Register 12: Allow Unregulated LDO to Reset the Part (LDOARST), offset 0x160

This register is provided as a means of allowing the LDO to reset the part if the voltage goes unregulated. Use this register to choose whether to automatically reset the part if the LDO goes unregulated, based on the design tolerance for LDO fluctuation.

Allow Unregulated LDO to Reset the Part (LDOARST)

Base 0x400F.E000
Offset 0x160
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															LDOARST
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:1	Reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	LDOARST	R/W	0	LDO Reset When set, allows unregulated LDO output to reset the part.

Register 13: 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	0	0	0	0	0	0	0	1	0	1	0	1	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								TEMP			PKG		ROHS	QUAL	
Type	RO	RO	RO	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	1	1	-	-

Bit/Field	Name	Type	Reset	Description
31:28	VER	RO	0x0	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 0x0 Initial **DID1** register format definition, indicating a Stellaris LM3Snnn device.
27:24	FAM	RO	0x0	Family This field provides the family identification of the device within the Luminary Micro product portfolio. The value is encoded as follows (all other encodings are reserved): Value Description 0x0 Stellaris family of microcontollers, that is, all devices with external part numbers starting with LM3S.
23:16	PARTNO	RO	0x2A	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 0x2A LM3S600
15:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
7:5	TEMP	RO	0x1	Temperature Range This field specifies the temperature rating of the device. The value is encoded as follows (all other encodings are reserved): Value Description 0x1 Industrial temperature range (-40°C to 85°C)
4:3	PKG	RO	0x1	Package Type This field specifies the package type. The value is encoded as follows (all other encodings are reserved): Value Description 0x1 48-pin LQFP package
2	ROHS	RO	1	RoHS-Compliance This bit specifies whether the device is RoHS-compliant. A 1 indicates the part is RoHS-compliant.
1:0	QUAL	RO	-	Qualification Status This field specifies the qualification status of the device. The value is encoded as follows (all other encodings are reserved): Value Description 0x0 Engineering Sample (unqualified) 0x1 Pilot Production (unqualified) 0x2 Fully Qualified

Register 14: 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. Value Description 0x001F 8 KB of SRAM
15:0	FLASHSZ	RO	0x000F	Flash Size Indicates the size of the on-chip flash memory. Value Description 0x000F 32 KB of Flash

Register 15: Device Capabilities 1 (DC1), offset 0x010

This register provides a list of features available in the system. The Stellaris family uses this register format to indicate the availability of the following family features in the specific device: PWM, ADC, Watchdog timer, and debug capabilities. This register also indicates the maximum clock frequency and maximum ADC sample rate. The format of this register is consistent with the **RCGC0**, **SCGC0**, and **DCGC0** clock control registers and the **SRCR0** software reset control register.

Device Capabilities 1 (DC1)

Base 0x400F.E000
Offset 0x010
Type RO, reset 0x0000.309F

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	MINSYSDIV				reserved				MPU	reserved		PLL	WDT	SWO	SWD	JTAG
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	1	1	0	0	0	0	1	0	0	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:12	MINSYSDIV	RO	0x3	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 0x3 Specifies a 50-MHz CPU clock with a PLL divider of 4.
11:8	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	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.
6:5	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	PLL	RO	1	PLL Present When set, indicates that the on-chip Phase Locked Loop (PLL) is present.
3	WDT	RO	1	Watchdog Timer Present When set, indicates that a watchdog timer is present.

Bit/Field	Name	Type	Reset	Description
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 16: Device Capabilities 2 (DC2), offset 0x014

This register provides a list of features available in the system. The Stellaris family uses this register format to indicate the availability of the following family features in the specific device: Analog Comparators, General-Purpose Timers, I2Cs, QEIs, SSIs, and UARTs. The format of this register is consistent with the **RCGC1**, **SCGC1**, and **DCGC1** clock control registers and the **SRCR1** software reset control register.

Device Capabilities 2 (DC2)

Base 0x400F.E000
Offset 0x014
Type RO, reset 0x0707.1013

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved					COMP2	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	1	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			I2C0	reserved							SSI0	reserved		UART1	UART0
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	1	0	0	1	1

Bit/Field	Name	Type	Reset	Description
31:27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	COMP2	RO	1	Analog Comparator 2 Present When set, indicates that analog comparator 2 is present.
25	COMP1	RO	1	Analog Comparator 1 Present When set, indicates that analog comparator 1 is present.
24	COMP0	RO	1	Analog Comparator 0 Present When set, indicates that analog comparator 0 is present.
23:19	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 2 Present When set, indicates that General-Purpose Timer module 2 is present.
17	TIMER1	RO	1	Timer 1 Present When set, indicates that General-Purpose Timer module 1 is present.
16	TIMER0	RO	1	Timer 0 Present When set, indicates that General-Purpose Timer module 0 is present.
15:13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	RO	1	I2C Module 0 Present When set, indicates that I2C module 0 is present.

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

Register 17: Device Capabilities 3 (DC3), offset 0x018

This register provides a list of features available in the system. The Stellaris family uses this register format to indicate the availability of the following family features in the specific device: Analog Comparator I/Os, CCP I/Os, ADC I/Os, and PWM I/Os.

Device Capabilities 3 (DC3)

Base 0x400F.E000
Offset 0x018
Type RO, reset 0x3F00.7FC0

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved		CCP5	CCP4	CCP3	CCP2	CCP1	CCP0	reserved							
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	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	C2O	C2PLUS	C2MINUS	C1O	C1PLUS	C1MINUS	C0O	C0PLUS	C0MINUS	reserved					
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:30	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
29	CCP5	RO	1	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.
23:15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	C2O	RO	1	C2o Pin Present When set, indicates that the analog comparator 2 output pin is present.
13	C2PLUS	RO	1	C2+ Pin Present When set, indicates that the analog comparator 2 (+) input pin is present.

Bit/Field	Name	Type	Reset	Description
12	C2MINUS	RO	1	C2- Pin Present When set, indicates that the analog comparator 2 (-) input pin is present.
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 18: Device Capabilities 4 (DC4), offset 0x01C

This register provides a list of features available in the system. The Stellaris family uses this register format to indicate the availability of GPIOs in the specific device. The format of this register is consistent with the **RCGC2**, **SCGC2**, and **DCGC2** clock control registers and the **SRCR2** software reset control register.

Device Capabilities 4 (DC4)

Base 0x400F.E000
Offset 0x01C
Type RO, reset 0x0000.001F

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

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

Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	GPIOE	RO	1	GPIO Port E Present When set, indicates that GPIO Port E is present.
3	GPIOD	RO	1	GPIO Port D Present When set, indicates that GPIO Port D is present.
2	GPIOC	RO	1	GPIO Port C Present When set, indicates that GPIO Port C is present.
1	GPIOB	RO	1	GPIO Port B Present When set, indicates that GPIO Port B is present.
0	GPIOA	RO	1	GPIO Port A Present When set, indicates that GPIO Port A is present.

Register 19: Run Mode Clock Gating Control Register 0 (RCGC0), offset 0x100

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

Run Mode Clock Gating Control Register 0 (RCGC0)

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

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

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT	R/W	0	WDT Clock Gating Control This bit controls the clock gating for the WDT module. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, a read or write to the unit generates a bus fault.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 20: Sleep Mode Clock Gating Control Register 0 (SCGC0), offset 0x110

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

Sleep Mode Clock Gating Control Register 0 (SCGC0)

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

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

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT	R/W	0	WDT Clock Gating Control This bit controls the clock gating for the WDT module. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, a read or write to the unit generates a bus fault.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 21: Deep Sleep Mode Clock Gating Control Register 0 (DCGC0), offset 0x120

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

Deep Sleep Mode Clock Gating Control Register 0 (DCGC0)

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

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

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT	R/W	0	WDT Clock Gating Control This bit controls the clock gating for the WDT module. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, a read or write to the unit generates a bus fault.
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 22: Run Mode Clock Gating Control Register 1 (RCGC1), offset 0x104

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

Run Mode Clock Gating Control Register 1 (RCGC1)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved					COMP2	COMP1	COMP0	reserved					TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	R/W	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			I2C0	reserved							SSI0	reserved		UART1	UART0
Type	RO	RO	RO	R/W	RO	RO	RO	RO	RO	RO	RO	R/W	RO	RO	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	COMP2	R/W	0	Analog Comparator 2 Clock Gating This bit controls the clock gating for analog comparator 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
25	COMP1	R/W	0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
23:19	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	TIMER2	R/W	0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
17	TIMER1	R/W	0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
15:13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Clock Gating Control This bit controls the clock gating for I2C module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	R/W	0	SSI0 Clock Gating Control This bit controls the clock gating for SSI module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
3:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	UART1	R/W	0	UART1 Clock Gating Control This bit controls the clock gating for UART module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control This bit controls the clock gating for UART module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

Register 23: Sleep Mode Clock Gating Control Register 1 (SCGC1), offset 0x114

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

Sleep Mode Clock Gating Control Register 1 (SCGC1)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved					COMP2	COMP1	COMP0	reserved					TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	R/W	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			I2C0	reserved							SSI0	reserved		UART1	UART0
Type	RO	RO	RO	R/W	RO	RO	RO	RO	RO	RO	RO	R/W	RO	RO	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	COMP2	R/W	0	Analog Comparator 2 Clock Gating This bit controls the clock gating for analog comparator 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
25	COMP1	R/W	0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
23:19	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	TIMER2	R/W	0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
17	TIMER1	R/W	0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
15:13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Clock Gating Control This bit controls the clock gating for I2C module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	R/W	0	SSI0 Clock Gating Control This bit controls the clock gating for SSI module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
3:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	UART1	R/W	0	UART1 Clock Gating Control This bit controls the clock gating for UART module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control This bit controls the clock gating for UART module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

Register 24: Deep Sleep Mode Clock Gating Control Register 1 (DCGC1), offset 0x124

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

Deep Sleep Mode Clock Gating Control Register 1 (DCGC1)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			reserved			COMP2	COMP1	COMP0			reserved			TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	R/W	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		I2C0				reserved				SSI0	reserved		UART1	UART0
Type	RO	RO	RO	R/W	RO	RO	RO	RO	RO	RO	RO	R/W	RO	RO	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26	COMP2	R/W	0	Analog Comparator 2 Clock Gating This bit controls the clock gating for analog comparator 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
25	COMP1	R/W	0	Analog Comparator 1 Clock Gating This bit controls the clock gating for analog comparator 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
24	COMP0	R/W	0	Analog Comparator 0 Clock Gating This bit controls the clock gating for analog comparator 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
23:19	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	TIMER2	R/W	0	Timer 2 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 2. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
17	TIMER1	R/W	0	Timer 1 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
16	TIMER0	R/W	0	Timer 0 Clock Gating Control This bit controls the clock gating for General-Purpose Timer module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
15:13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Clock Gating Control This bit controls the clock gating for I2C module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
11:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	SSI0	R/W	0	SSI0 Clock Gating Control This bit controls the clock gating for SSI module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
3:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	UART1	R/W	0	UART1 Clock Gating Control This bit controls the clock gating for UART module 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
0	UART0	R/W	0	UART0 Clock Gating Control This bit controls the clock gating for UART module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

Register 25: Run Mode Clock Gating Control Register 2 (RCGC2), offset 0x108

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

Run Mode Clock Gating Control Register 2 (RCGC2)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
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	0	Software should not rely on the value of 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 This bit controls the clock gating for Port E. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
3	GPIOD	R/W	0	Port D Clock Gating Control This bit controls the clock gating for Port D. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
2	GPIOC	R/W	0	Port C Clock Gating Control This bit controls the clock gating for Port C. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
1	GPIOB	R/W	0	Port B Clock Gating Control This bit controls the clock gating for Port B. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

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

Register 26: Sleep Mode Clock Gating Control Register 2 (SCGC2), offset 0x118

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

Sleep Mode Clock Gating Control Register 2 (SCGC2)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
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	0	Software should not rely on the value of 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 This bit controls the clock gating for Port E. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
3	GPIOD	R/W	0	Port D Clock Gating Control This bit controls the clock gating for Port D. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
2	GPIOC	R/W	0	Port C Clock Gating Control This bit controls the clock gating for Port C. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
1	GPIOB	R/W	0	Port B Clock Gating Control This bit controls the clock gating for Port B. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

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

Register 27: Deep Sleep Mode Clock Gating Control Register 2 (DCGC2), offset 0x128

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

Deep Sleep Mode Clock Gating Control Register 2 (DCGC2)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
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	0	Software should not rely on the value of 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 This bit controls the clock gating for Port E. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
3	GPIOD	R/W	0	Port D Clock Gating Control This bit controls the clock gating for Port D. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
2	GPIOC	R/W	0	Port C Clock Gating Control This bit controls the clock gating for Port C. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
1	GPIOB	R/W	0	Port B Clock Gating Control This bit controls the clock gating for Port B. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.

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

Register 28: Software Reset Control 0 (SRCR0), offset 0x040

Writes to this register are masked by the bits in the **Device Capabilities 1 (DC1)** register.

Software Reset Control 0 (SRCR0)

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

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

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT	R/W	0	WDT Reset Control Reset control for Watchdog unit.
2:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 29: Software Reset Control 1 (SRCR1), offset 0x044

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

Software Reset Control 1 (SRCR1)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved					COMP2	COMP1	COMP0	reserved					TIMER2	TIMER1	TIMER0
Type	RO	RO	RO	RO	RO	R/W	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			I2C0	reserved							SSI0	reserved		UART1	UART0
Type	RO	RO	RO	R/W	RO	RO	RO	RO	RO	RO	RO	R/W	RO	RO	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
4	SSI0	R/W	0	SSI0 Reset Control Reset control for SSI unit 0.
3:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	UART1	R/W	0	UART1 Reset Control Reset control for UART unit 1.
0	UART0	R/W	0	UART0 Reset Control Reset control for UART unit 0.

Register 30: Software Reset Control 2 (SRCR2), offset 0x048

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

Software Reset Control 2 (SRCR2)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
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

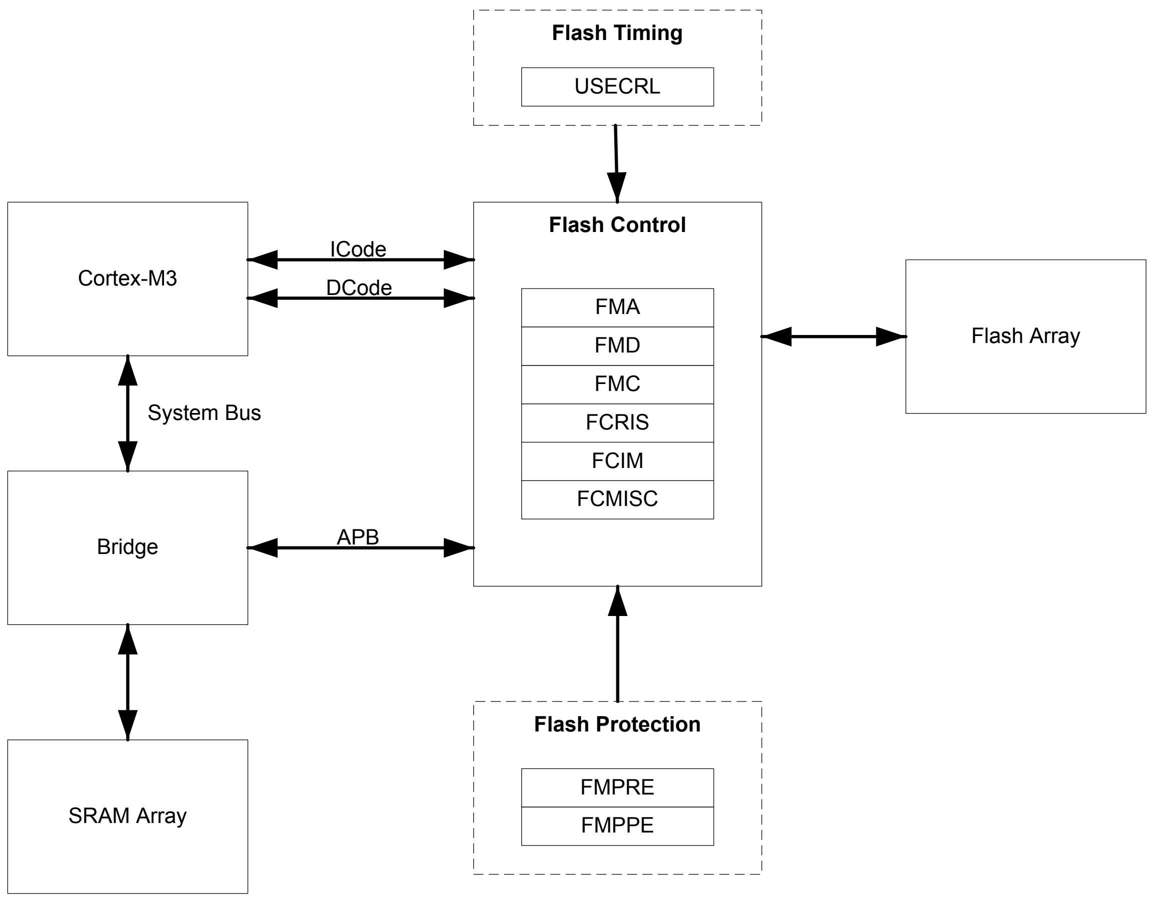
Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	GPIOE	R/W	0	Port E Reset Control Reset control for GPIO Port E.
3	GPIOD	R/W	0	Port D Reset Control Reset control for GPIO Port D.
2	GPIOC	R/W	0	Port C Reset Control Reset control for GPIO Port C.
1	GPIOB	R/W	0	Port B Reset Control Reset control for GPIO Port B.
0	GPIOA	R/W	0	Port A Reset Control Reset control for GPIO Port A.

7 Internal Memory

The LM3S600 microcontroller comes with 8 KB of bit-banded SRAM and 32 KB of flash memory. The flash controller provides a user-friendly interface, making flash programming a simple task. Flash protection can be applied to the flash memory on a 2-KB block basis.

7.1 Block Diagram

Figure 7-1. Flash Block Diagram



7.2 Functional Description

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

7.2.1 SRAM Memory

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

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

```
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*.

7.2.2 Flash Memory

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

See also “Serial Flash Loader” on page 360 for a preprogrammed flash-resident utility used to download code to the flash memory of a device without the use of a debug interface.

7.2.2.1 Flash Memory Timing

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

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

7.2.2.2 Flash Memory Protection

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

- **Flash Memory Protection Program Enable (FMPPEn)**: If set, the block may be programmed (written) or erased. If cleared, the block may not be changed.

- **Flash Memory Protection Read Enable (FMPREn)**: If set, the block may be executed or read by software or debuggers. If cleared, the block may only be executed. The contents of the memory block are prohibited from being accessed as data and traversing the DCode bus.

The policies may be combined as shown in Table 7-1 on page 105.

Table 7-1. Flash Protection Policy Combinations

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

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

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

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

7.2.2.3 Flash Protection by Disabling Debug Access

Flash memory may also be protected by permanently disabling access to the Debug Access Port (DAP) through the JTAG and SWD interfaces. This is accomplished by clearing the DBG field of the **FMPRE** register.

Flash Memory Protection Read Enable (DBG field): If set to 0x2, access to the DAP is enabled through the JTAG and SWD interfaces. If clear, access to the DAP is disabled. The DBG field programming becomes permanent, and irreversible, after a commit sequence is performed.

In the initial state, provided from the factory, access is enabled in order to facilitate code development and debug. Access to the DAP may be disabled at the end of the manufacturing flow, once all tests have passed and software loaded. This change will not take effect until the next power-up of the device. Note that it is recommended that disabling access to the DAP be combined with a mechanism for providing end-user installable updates (if necessary) such as the Stellaris boot loader.

Important: Once the DBG field is cleared and committed, this field can never be restored to the factory-programmed value—which means JTAG/SWD interface to the debug module can never be re-enabled. This sequence does NOT disable the JTAG controller, it only disables the access of the DAP through the JTAG or SWD interfaces. The JTAG interface remains functional and access to the Test Access Port remains enabled, allowing the user to execute the IEEE JTAG-defined instructions (for example, to perform boundary scan operations).

If the user will also be using the **FMPRE** bits to protect flash memory from being read as data (to mark sets of 2 KB blocks of flash memory as execute-only), these one-time-programmable bits should be written at the same time that the debug disable bits are programmed. Mechanisms to execute the one-time code sequence to disable all debug access include:

- Selecting the debug disable option in the Stellaris boot loader

- Loading the debug disable sequence into SRAM and running it once from SRAM after programming the final end application code into flash

7.3 Flash Memory Initialization and Configuration

This section shows examples for using the flash controller to perform various operations on the contents of the flash memory.

7.3.1 Changing Flash Protection Bits

As discussed in "Flash Memory Protection" on page 104, changes to the protection bits must be committed before they take effect. The sequence below is used change and commit a block protection bit in the **FMPRE** or **FMPPE** registers. The sequence to change and commit a bit in software is as follows:

1. The **Flash Memory Protection Read Enable (FMPRE)** and **Flash Memory Protection Program Enable (FMPPE)** registers are written, changing the intended bit(s). The action of these changes can be tested by software while in this state.
2. The **Flash Memory Address (FMA)** register (see page 109) bit 0 is set to 1 if the **FMPPE** register is to be committed; otherwise, a 0 commits the **FMPRE** register.
3. The **Flash Memory Control (FMC)** register (see page 111) is written with the `COMT` bit set. This initiates a write sequence and commits the changes.

There is a special sequence to change and commit the `DBG` bits in the **Flash Memory Protection Read Enable (FMPRE)** register. This sequence also sets and commits any changes from 1 to 0 in the block protection bits (for execute-only) in the **FMPRE** register.

1. The F**lash Memory Protection Read Enable (FMPRE)** register is written, changing the intended bit(s). The action of these changes can be tested by software while in this state.
2. The **Flash Memory Address (FMA)** register (see ppage 109) is written with a value of 0x900.
3. The **Flash Memory Control (FMC)** register (see page 111) is written with the `COMT` bit set. This initiates a write sequence and commits the changes.

Below is an example code sequence to permanently disable the JTAG and SWD interface to the debug module using Luminary Micro's DriverLib peripheral driver library:

```
#include "hw_types.h"
#include "hw_flash.h"
void
permanently_disable_jtag_swd(void)
{
     //
     // Clear the DBG field of the FMPRE register. Note that the value
     // used in this instance does not affect the state of the BlockN
     // bits, but were the value different, all bits in the FMPRE are
     // affected by this function!
     //
     HWREG(FLASH_FMPRE) &= 0x3fffffff;
     //
     // The following sequence activates the one-time
```

```
        // programming of the FMPRE register.
        //
        HWREG(FLASH_FMA) = 0x900;
        HWREG(FLASH_FMC) = (FLASH_FMC_WRKEY | FLASH_FMC_COMT);
        //
        // Wait until the operation is complete.
        //
        while (HWREG(FLASH_FMC) & FLASH_FMC_COMT)
        {
        }
}
```

7.3.2 Flash Programming

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

7.3.2.1 To program a 32-bit word

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

7.3.2.2 To perform an erase of a 1-KB page

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

7.3.2.3 To perform a mass erase of the flash

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

7.4 Register Map

Table 7-2 on page 108 lists the Flash memory and control registers. The offset listed is a hexadecimal increment to the register's address. The **FMA**, **FMD**, **FMC**, **FCRIS**, **FCIM**, and **FCMISC** registers are relative to the Flash control base address of 0x400F.D000. The **FMPREn**, **FMPPEn**, **USECRL**, **USER_DBG**, and **USER_REGn** registers are relative to the System Control base address of 0x400F.E000.

Table 7-2. Flash Register Map

Offset	Name	Type	Reset	Description	See page
Flash Control Offset					
0x000	FMA	R/W	0x0000.0000	Flash Memory Address	109
0x004	FMD	R/W	0x0000.0000	Flash Memory Data	110
0x008	FMC	R/W	0x0000.0000	Flash Memory Control	111
0x00C	FCRIS	RO	0x0000.0000	Flash Controller Raw Interrupt Status	113
0x010	FCIM	R/W	0x0000.0000	Flash Controller Interrupt Mask	114
0x014	FCMISC	R/W1C	0x0000.0000	Flash Controller Masked Interrupt Status and Clear	115
System Control Offset					
0x130	FMPRE	R/W	0x8000.FFFF	Flash Memory Protection Read Enable	117
0x134	FMPPE	R/W	0x0000.FFFF	Flash Memory Protection Program Enable	118
0x140	USECRL	R/W	0x31	USec Reload	116

7.5 Flash Register Descriptions (Flash Control Offset)

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

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

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

Flash Memory Address (FMA)

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

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

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

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

Flash Memory Data (FMD)

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

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

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

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

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

When this register is written, the flash controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 109). If the access is a write access, the data contained in the **Flash Memory Data (FMD)** register (see page 110) is written.

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

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

Flash Memory Control (FMC)

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

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

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

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

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

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

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

Flash Controller Raw Interrupt Status (FCRIS)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														PRIS	ARIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
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	PRIS	RO	0	Programming Raw Interrupt Status This bit indicates the current state of the programming cycle. If set, the programming cycle completed; if cleared, the programming cycle has not completed. Programming cycles are either write or erase actions generated through the **Flash Memory Control (FMC)** register bits (see page 111).
0	ARIS	RO	0	Access Raw Interrupt Status This bit indicates if the flash was improperly accessed. If set, the program tried to access the flash counter to the policy as set in the **Flash Memory Protection Read Enable (FMPREn)** and **Flash Memory Protection Program Enable (FMPPEn)** registers. Otherwise, no access has tried to improperly access the flash.

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

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

Flash Controller Interrupt Mask (FCIM)

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PMASK	R/W	0	Programming Interrupt Mask This bit controls the reporting of the programming raw interrupt status to the controller. If set, a programming-generated interrupt is promoted to the controller. Otherwise, interrupts are recorded but suppressed from the controller.
0	AMASK	R/W	0	Access Interrupt Mask This bit controls the reporting of the access raw interrupt status to the controller. If set, an access-generated interrupt is promoted to the controller. Otherwise, interrupts are recorded but suppressed from the controller.

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

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

Flash Controller Masked Interrupt Status and Clear (FCMISC)

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

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

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

Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PMISC	R/W1C	0	Programming Masked Interrupt Status and Clear This bit indicates whether an interrupt was signaled because a programming cycle completed and was not masked. This bit is cleared by writing a 1. The `PRIS` bit in the **FCRIS** register (see page 113) is also cleared when the `PMISC` bit is cleared.
0	AMISC	R/W1C	0	Access Masked Interrupt Status and Clear This bit indicates whether an interrupt was signaled because an improper access was attempted and was not masked. This bit is cleared by writing a 1. The `ARIS` bit in the **FCRIS** register is also cleared when the `AMISC` bit is cleared.

7.6 Flash Register Descriptions (System Control Offset)

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

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

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

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

USec Reload (USECRL)

Base 0x400F.E000
Offset 0x140
Type R/W, reset 0x31

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

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	USEC	R/W	0x31	Microsecond Reload Value MHz -1 of the controller clock when the flash is being erased or programmed. USEC should be set to 0x31 (50 MHz) whenever the flash is being erased or programmed.

Register 8: Flash Memory Protection Read Enable (FMPRE), offset 0x130

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

Flash Memory Protection Read Enable (FMPRE)

Base 0x400F.E000
Offset 0x130
Type R/W, reset 0x8000.FFFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	READ_ENABLE															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	0	0	0	0	0	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	0x8000FFFF	Flash Read Enable Each bit position maps 2 Kbytes of Flash to be read-enabled.

Value	Description
0x8000FFFF	Enables 32 KB of flash.

Register 9: Flash Memory Protection Program Enable (FMPPE), offset 0x134

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

Flash Memory Protection Program Enable (FMPPE)

Base 0x400F.E000
Offset 0x134
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 Each bit position maps 2 Kbytes of Flash to be write-enabled. Value: 0x0000FFFF — Description: Enables 32 KB of flash.

8 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, and Port E,). The GPIO module is FiRM-compliant and supports 8-36 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

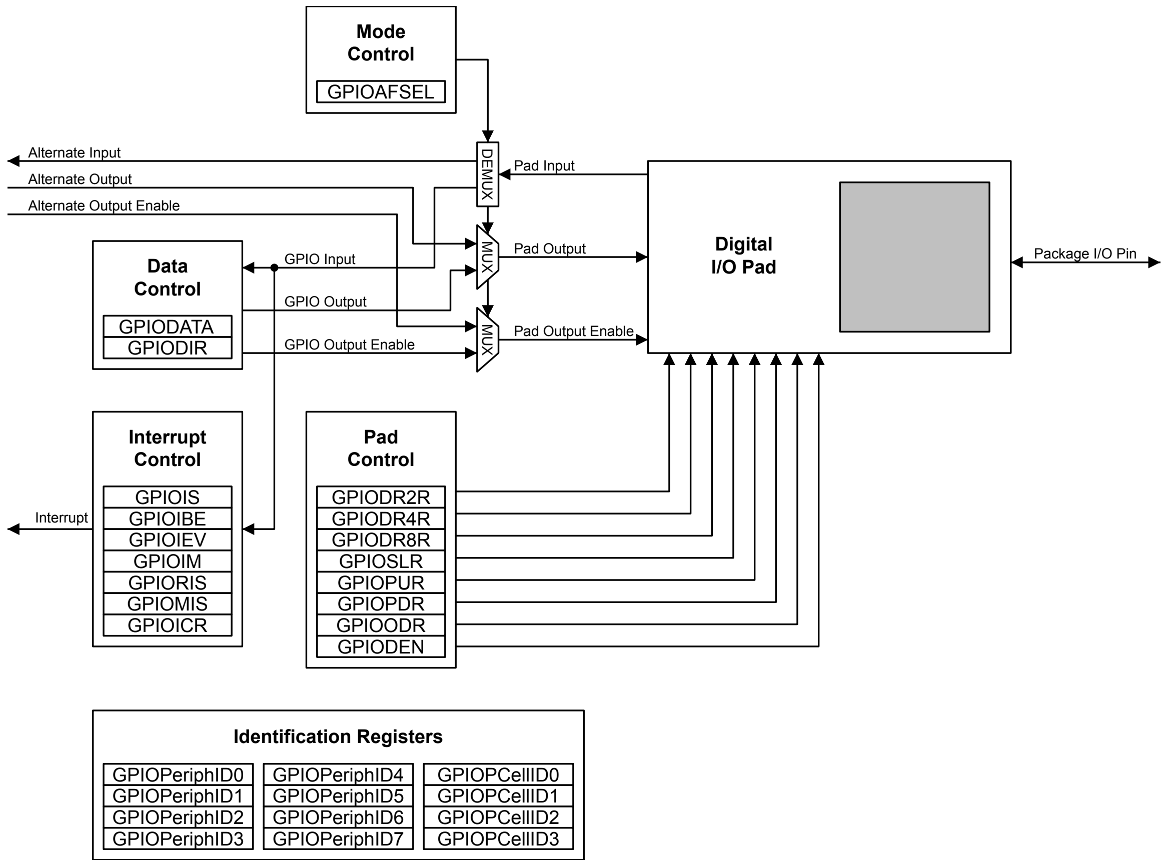
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
- 5-V-tolerant input/outputs
- Bit masking in both read and write operations through address lines
- Programmable control for GPIO pad configuration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

8.1 Functional Description

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

Each GPIO port is a separate hardware instantiation of the same physical block (see Figure 8-1 on page 120). The LM3S600 microcontroller contains five ports and thus five of these physical GPIO blocks.

Figure 8-1. GPIO Port Block Diagram



8.1.1 Data Control

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

8.1.1.1 Data Direction Operation

The **GPIO Direction (GPIODIR)** register (see page 127) is used to configure each individual pin as an input or output. When the data direction bit is set to 0, the GPIO is configured as an input and the corresponding data register bit will capture and store the value on the GPIO port. When the data direction bit is set to 1, the GPIO is configured as an output and the corresponding data register bit will be driven out on the GPIO port.

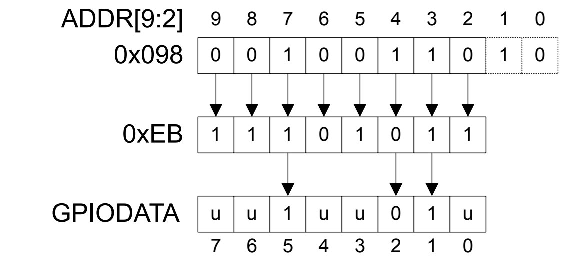
8.1.1.2 Data Register Operation

To aid in the efficiency of software, the GPIO ports allow for the modification of individual bits in the **GPIO Data (GPIODATA)** register (see page 126) by using bits [9:2] of the address bus as a mask. This allows software drivers to modify individual GPIO pins in a single instruction, without affecting the state of the other pins. This is in contrast to the "typical" method of doing a read-modify-write operation to set or clear an individual GPIO pin. To accommodate this feature, the **GPIODATA** register covers 256 locations in the memory map.

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

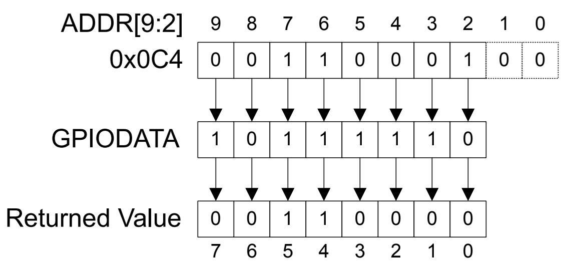
For example, writing a value of 0xEB to the address GPIODATA + 0x098 would yield as shown in Figure 8-2 on page 121, where u is data unchanged by the write.

Figure 8-2. GPIODATA Write Example



During a read, if the address bit associated with the data bit is set to 1, the value is read. If the address bit associated with the data bit is set to 0, it is read as a zero, regardless of its actual value. For example, reading address GPIODATA + 0x0C4 yields as shown in Figure 8-3 on page 121.

Figure 8-3. GPIODATA Read Example



8.1.2 Interrupt Control

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

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

- **GPIO Interrupt Sense (GPIOIS)** register (see page 128)
- **GPIO Interrupt Both Edges (GPIOIBE)** register (see page 129)
- **GPIO Interrupt Event (GPIOIEV)** register (see page 130)

Interrupts are enabled/disabled via the **GPIO Interrupt Mask (GPIOIM)** register (see page 131).

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 132 and page 133). As the name implies, the **GPIOMIS** register only shows interrupt conditions that are allowed to be passed to the controller. The **GPIORIS** register indicates that a GPIO pin meets the conditions for an interrupt, but has not necessarily been sent to the controller.

Interrupts are cleared by writing a 1 to the **GPIO Interrupt Clear (GPIOICR)** register (see page 134).

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

8.1.3 Mode Control

The GPIO pins can be controlled by either hardware or software. When hardware control is enabled via the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 135), the pin state is controlled by its alternate function (that is, the peripheral). Software control corresponds to GPIO mode, where the **GPIODATA** register is used to read/write the corresponding pins.

8.1.4 Pad Control

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

8.1.5 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.

8.2 Initialization and Configuration

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

On reset, all GPIO pins (except for the five JTAG pins) default to general-purpose inut mode (**GPIODIR**=0 and **GPIOAFSEL**=0). Table 8-1 on page 122 shows all possible configurations of the GPIO pads and the control register settings required to achieve them. Table 8-2 on page 123 shows how a rising edge interrupt would be configured for pin 2 of a GPIO port.

Table 8-1. GPIO Pad Configuration Examples

Configuration	GPIO Register Bit Value[a]									
	AFSEL	DIR	ODR	DEN	PUR	PDR	DR2R	DR4R	DR8R	SLR
Digital Input (GPIO)	0	0	0	1	?	?	X	X	X	X
Digital Output (GPIO)	0	1	0	1	?	?	?	?	?	?
Open Drain Input (GPIO)	0	0	1	1	X	X	X	X	X	X
Open Drain Output (GPIO)	0	1	1	1	X	X	?	?	?	?
Open Drain Input/Output (I^2C)	1	X	1	1	X	X	?	?	?	?
Digital Input (Timer CCP)	1	X	0	1	?	?	X	X	X	X
Digital Output (Timer PWM)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (SSI)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (UART)	1	X	0	1	?	?	?	?	?	?

Configuration	GPIO Register Bit Value[a]									
	AFSEL	DIR	ODR	DEN	PUR	PDR	DR2R	DR4R	DR8R	SLR
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 8-2. GPIO Interrupt Configuration Example

Register	Desired Interrupt Event Trigger	Pin 2 Bit Value[a]							
		7	6	5	4	3	2	1	0
GPIOIS	0=edge 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 negative edge 1=High level, or positive edge	X	X	X	X	X	1	X	X
GPIOIM	0=masked 1=not masked	0	0	0	0	0	1	0	0

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

8.3 Register Map

Table 8-3 on page 124 lists the GPIO registers. The offset listed is a hexadecimal increment to the register's address, relative to that GPIO port's base address:

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

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

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

Table 8-3. GPIO Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPIODATA	R/W	0x0000.0000	GPIO Data	126
0x400	GPIODIR	R/W	0x0000.0000	GPIO Direction	127
0x404	GPIOIS	R/W	0x0000.0000	GPIO Interrupt Sense	128
0x408	GPIOIBE	R/W	0x0000.0000	GPIO Interrupt Both Edges	129
0x40C	GPIOIEV	R/W	0x0000.0000	GPIO Interrupt Event	130
0x410	GPIOIM	R/W	0x0000.0000	GPIO Interrupt Mask	131
0x414	GPIORIS	RO	0x0000.0000	GPIO Raw Interrupt Status	132
0x418	GPIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status	133
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear	134
0x420	GPIOAFSEL	R/W	-	GPIO Alternate Function Select	135
0x500	GPIODR2R	R/W	0x0000.00FF	GPIO 2-mA Drive Select	137
0x504	GPIODR4R	R/W	0x0000.0000	GPIO 4-mA Drive Select	138
0x508	GPIODR8R	R/W	0x0000.0000	GPIO 8-mA Drive Select	139
0x50C	GPIOODR	R/W	0x0000.0000	GPIO Open Drain Select	140
0x510	GPIOPUR	R/W	0x0000.00FF	GPIO Pull-Up Select	141
0x514	GPIOPDR	R/W	0x0000.0000	GPIO Pull-Down Select	142
0x518	GPIOSLR	R/W	0x0000.0000	GPIO Slew Rate Control Select	143
0x51C	GPIODEN	R/W	0x0000.00FF	GPIO Digital Enable	144
0xFD0	GPIOPeriphID4	RO	0x0000.0000	GPIO Peripheral Identification 4	145
0xFD4	GPIOPeriphID5	RO	0x0000.0000	GPIO Peripheral Identification 5	146
0xFD8	GPIOPeriphID6	RO	0x0000.0000	GPIO Peripheral Identification 6	147
0xFDC	GPIOPeriphID7	RO	0x0000.0000	GPIO Peripheral Identification 7	148
0xFE0	GPIOPeriphID0	RO	0x0000.0061	GPIO Peripheral Identification 0	149
0xFE4	GPIOPeriphID1	RO	0x0000.0000	GPIO Peripheral Identification 1	150
0xFE8	GPIOPeriphID2	RO	0x0000.0018	GPIO Peripheral Identification 2	151
0xFEC	GPIOPeriphID3	RO	0x0000.0001	GPIO Peripheral Identification 3	152
0xFF0	GPIOPCellID0	RO	0x0000.000D	GPIO PrimeCell Identification 0	153
0xFF4	GPIOPCellID1	RO	0x0000.00F0	GPIO PrimeCell Identification 1	154
0xFF8	GPIOPCellID2	RO	0x0000.0005	GPIO PrimeCell Identification 2	155

Offset	Name	Type	Reset	Description	See page
0xFFC	GPIOPCellID3	RO	0x0000.00B1	GPIO PrimeCell Identification 3	156

8.4 Register Descriptions

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

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

The **GPIODATA** register is the data register. In software control mode, values written in the **GPIODATA** register are transferred onto the GPIO port pins if the respective pins have been configured as outputs through the **GPIO Direction (GPIODIR)** register (see page 127).

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

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

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

GPIO Data (GPIODATA)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved 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 the data written to the registers are masked by the eight address lines `ipaddr[9:2]`. Reads from this register return its current state. Writes to this register only affect bits that are not masked by `ipaddr[9:2]` and are configured as outputs. See “Data Register Operation” on page 120 for examples of reads and writes.

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

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

GPIO Direction (GPIODIR)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIR	R/W	0x00	GPIO Data Direction The `DIR` values are defined as follows: Value Description 0 Pins are inputs. 1 Pins are outputs.

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

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

GPIO Interrupt Sense (GPIOIS)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IS	R/W	0x00	GPIO Interrupt Sense The `IS` values are defined as follows: Value Description 0 Edge on corresponding pin is detected (edge-sensitive). 1 Level on corresponding pin is detected (level-sensitive).

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

The **GPIOIBE** register is the interrupt both-edges register. When the corresponding bit in the **GPIO Interrupt Sense (GPIOIS)** register (see page 128) is set to detect edges, bits set to High in **GPIOIBE** configure the corresponding pin to detect both rising and falling edges, regardless of the corresponding bit in the **GPIO Interrupt Event (GPIOIEV)** register (see page 130). Clearing a bit configures the pin to be controlled by **GPIOIEV**. All bits are cleared by a reset.

GPIO Interrupt Both Edges (GPIOIBE)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IBE	R/W	0x00	GPIO Interrupt Both Edges

The `IBE` values are defined as follows:

Value	Description
0	Interrupt generation is controlled by the **GPIO Interrupt Event (GPIOIEV)** register (see page 130).
1	Both edges on the corresponding pin trigger an interrupt.

Note: Single edge is determined by the corresponding bit in **GPIOIEV**.

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

The **GPIOIEV** register is the interrupt event register. Bits set to High in **GPIOIEV** configure the corresponding pin to detect rising edges or high levels, depending on the corresponding bit value in the **GPIO Interrupt Sense (GPIOIS)** register (see page 128). Clearing a bit configures the pin to detect falling edges or low levels, depending on the corresponding bit value in **GPIOIS**. All bits are cleared by a reset.

GPIO Interrupt Event (GPIOIEV)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IEV	R/W	0x00	GPIO Interrupt Event The `IEV` values are defined as follows: Value Description 0 Falling edge or Low levels on corresponding pins trigger interrupts. 1 Rising edge or High levels on corresponding pins trigger interrupts.

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

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

GPIO Interrupt Mask (GPIOIM)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IME	R/W	0x00	GPIO Interrupt Mask Enable The `IME` values are defined as follows: Value Description 0 Corresponding pin interrupt is masked. 1 Corresponding pin interrupt is not masked.

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

The **GPIORIS** register is the raw interrupt status register. Bits read High in **GPIORIS** reflect the status of interrupt trigger conditions detected (raw, prior to masking), indicating that all the requirements have been met, before they are finally allowed to trigger by the **GPIO Interrupt Mask (GPIOIM)** register (see page 131). Bits read as zero indicate that corresponding input pins have not initiated an interrupt. All bits are cleared by a reset.

GPIO Raw Interrupt Status (GPIORIS)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	RIS	RO	0x00	GPIO Interrupt Raw Status Reflects the status of interrupt trigger condition detection on pins (raw, prior to masking). The `RIS` values are defined as follows: Value Description 0 Corresponding pin interrupt requirements not met. 1 Corresponding pin interrupt has met requirements.

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

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

GPIOMIS is the state of the interrupt after masking.

GPIO Masked Interrupt Status (GPIOMIS)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	MIS	RO	0x00	GPIO Masked Interrupt Status Masked value of interrupt due to corresponding pin. The `MIS` values are defined as follows: Value Description 0 Corresponding GPIO line interrupt not active. 1 Corresponding GPIO line asserting interrupt.

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

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

GPIO Interrupt Clear (GPIOICR)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IC	W1C	0x00	GPIO Interrupt Clear The `IC` values are defined as follows: Value Description 0 Corresponding interrupt is unaffected. 1 Corresponding interrupt is cleared.

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

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

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

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

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

GPIO Alternate Function Select (GPIOAFSEL)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

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

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

GPIO 2-mA Drive Select (GPIODR2R)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV2	R/W	0xFF	Output Pad 2-mA Drive Enable A write of 1 to either **GPIODR4[n]** or **GPIODR8[n]** clears the corresponding 2-mA enable bit. The change is effective on the second clock cycle after the write.

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

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

GPIO 4-mA Drive Select (GPIODR4R)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV4	R/W	0x00	Output Pad 4-mA Drive Enable A write of 1 to either **GPIODR2[n]** or **GPIODR8[n]** clears the corresponding 4-mA enable bit. The change is effective on the second clock cycle after the write.

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

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

GPIO 8-mA Drive Select (GPIODR8R)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DRV8	R/W	0x00	Output Pad 8-mA Drive Enable A write of 1 to either **GPIODR2[n]** or **GPIODR4[n]** clears the corresponding 8-mA enable bit. The change is effective on the second clock cycle after the write.

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

The **GPIOODR** register is the open drain control register. Setting a bit in this register enables the open drain configuration of the corresponding GPIO pad. When open drain mode is enabled, the corresponding bit should also be set in the **GPIO Digital Input Enable (GPIODEN)** register (see page 144). Corresponding bits in the drive strength registers (**GPIODR2R**, **GPIODR4R**, **GPIODR8R**, and **GPIOSLR**) can be set to achieve the desired rise and fall times. The GPIO acts as an open drain input if the corresponding bit in the **GPIODIR** register is set to 0; and as an open drain output when set to 1.

When using the I^{2}C module, the **GPIO Alternate Function Select (GPIOAFSEL)** register bit for PB2 and PB3 should be set to 1 (see examples in “Initialization and Configuration” on page 122).

GPIO Open Drain Select (GPIOODR)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ODE	R/W	0x00	Output Pad Open Drain Enable The `ODE` values are defined as follows: Value Description 0 Open drain configuration is disabled. 1 Open drain configuration is enabled.

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

The **GPIOPUR** register is the pull-up control register. When a bit is set to 1, it enables a weak pull-up resistor on the corresponding GPIO signal. Setting a bit in **GPIOPUR** automatically clears the corresponding bit in the **GPIO Pull-Down Select (GPIOPDR)** register (see page 142).

GPIO Pull-Up Select (GPIOPUR)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
Offset 0x510
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								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	1	1	1	1	1	1	1	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	PUE	R/W	0xFF	Pad Weak Pull-Up Enable A write of 1 to **GPIOPDR[n]** clears the corresponding **GPIOPUR[n]** enables. The change is effective on the second clock cycle after the write.

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

The **GPIOPDR** register is the pull-down control register. When a bit is set to 1, it enables a weak pull-down resistor on the corresponding GPIO signal. Setting a bit in **GPIOPDR** automatically clears the corresponding bit in the **GPIO Pull-Up Select (GPIOPUR)** register (see page 141).

GPIO Pull-Down Select (GPIOPDR)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PDE	R/W	0x00	Pad Weak Pull-Down Enable A write of 1 to **GPIOPUR[n]** clears the corresponding **GPIOPDR[n]** enables. The change is effective on the second clock cycle after the write.

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

The **GPIOSLR** register is the slew rate control register. Slew rate control is only available when using the 8-mA drive strength option via the **GPIO 8-mA Drive Select (GPIODR8R)** register (see page 139).

GPIO Slew Rate Control Select (GPIOSLR)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	SRL	R/W	0x00	Slew Rate Limit Enable (8-mA drive only) The SRL values are defined as follows: Value Description 0 Slew rate control disabled. 1 Slew rate control enabled.

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

The **GPIODEN** register is the digital input enable register. By default, all GPIO signals are configured as digital inputs at reset. If a pin is being used as a GPIO or its Alternate Hardware Function, it should be configured as a digital input. The only time that a pin should not be configured as a digital input is when the GPIO pin is configured to be one of the analog input signals for the analog comparators.

GPIO Digital Enable (GPIODEN)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
Offset 0x51C
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								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	1	1	1	1	1	1	1	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	DEN	R/W	0xFF	Digital Enable The DEN values are defined as follows: Value Description 0 Digital functions disabled. 1 Digital functions enabled.

Register 19: GPIO Peripheral Identification 4 (GPIOPeriphID4), offset 0xFD0

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

GPIO Peripheral Identification 4 (GPIOPeriphID4)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
Offset 0xFD0
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Register 20: GPIO Peripheral Identification 5 (GPIOPeriphID5), offset 0xFD4

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

GPIO Peripheral Identification 5 (GPIOPeriphID5)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved 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 21: GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8

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

GPIO Peripheral Identification 6 (GPIOPeriphID6)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
Offset 0xFD8
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID6							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Register 22: GPIO Peripheral Identification 7 (GPIOPeriphID7), offset 0xFDC

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

GPIO Peripheral Identification 7 (GPIOPeriphID7)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID7							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Register 23: GPIO Peripheral Identification 0 (GPIOPeriphID0), offset 0xFE0

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

GPIO Peripheral Identification 0 (GPIOPeriphID0)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved 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 24: GPIO Peripheral Identification 1 (GPIOPeriphID1), offset 0xFE4

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

GPIO Peripheral Identification 1 (GPIOPeriphID1)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved 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 25: GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8

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

GPIO Peripheral Identification 2 (GPIOPeriphID2)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved 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 26: GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC

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

GPIO Peripheral Identification 3 (GPIOPeriphID3)

GPIO Port A base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved 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 27: 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	0x00	Software should not rely on the value of a reserved 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 28: 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
Offset 0xFF4
Type RO, reset 0x0000.00F0

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

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

Register 29: 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	GPIO PrimeCell ID Register[23:16] Provides software a standard cross-peripheral identification system.

Register 30: 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 base: 0x4000.4000
GPIO Port B base: 0x4000.5000
GPIO Port C base: 0x4000.6000
GPIO Port D base: 0x4000.7000
GPIO Port E base: 0x4002.4000
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	GPIO PrimeCell ID Register[31:24] Provides software a standard cross-peripheral identification system.

9 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 (Timer0, Timer1, and Timer 2). Each GPTM block provides two 16-bit timer/counters (referred to as TimerA and TimerB) that can be configured to operate independently as timers or event counters, or configured to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC).

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

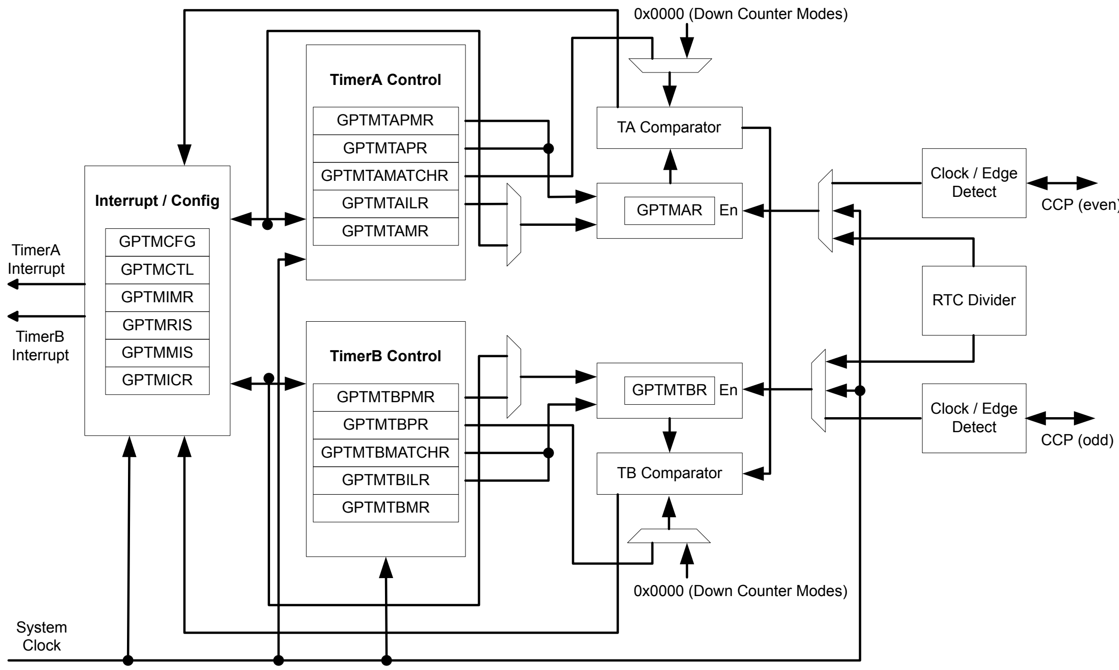
The General-Purpose Timer Module is one timing resource available on the Stellaris® microcontrollers. Other timer resources include the System Timer (SysTick) (see “System Timer (SysTick)” on page 31).

The following modes are supported:

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

9.1 Block Diagram

Figure 9-1. GPTM Module Block Diagram



9.2 Functional Description

The main components of each GPTM block are two free-running 16-bit up/down counters (referred to as TimerA and TimerB), two 16-bit match registers, two prescaler match registers, and two 16-bit load/initialization registers and their associated control functions. The exact functionality of each GPTM is controlled by software and configured through the register interface.

Software configures the GPTM using the **GPTM Configuration (GPTMCFG)** register (see page 169), the **GPTM TimerA Mode (GPTMTAMR)** register (see page 170), and the **GPTM TimerB Mode (GPTMTBMR)** register (see page 172). 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.

9.2.1 GPTM Reset Conditions

After reset has been applied to the GPTM module, the module is in an inactive state, and all control registers are cleared and in their default states. Counters TimerA and TimerB are initialized to 0xFFFF, along with their corresponding load registers: the **GPTM TimerA Interval Load (GPTMTAILR)** register (see page 183) and the **GPTM TimerB Interval Load (GPTMTBILR)** register (see page 184). The prescale counters are initialized to 0x00: the **GPTM TimerA Prescale (GPTMTAPR)** register (see page 187) and the **GPTM TimerB Prescale (GPTMTBPR)** register (see page 188).

9.2.2 32-Bit Timer Operating Modes

Note: Both the odd- and even-numbered CCP pins are used for 16-bit mode. Only the even-numbered CCP pins are used for 32-bit mode.

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

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

- **GPTM TimerA Interval Load (GPTMTAILR)** register [15:0], see page 183
- **GPTM TimerB Interval Load (GPTMTBILR)** register [15:0], see page 184
- **GPTM TimerA (GPTMTAR)** register [15:0], see page 191
- **GPTM TimerB (GPTMTBR)** register [15:0], see page 192

In the 32-bit modes, the GPTM translates a 32-bit write access to **GPTMTAILR** into a write access to both **GPTMTAILR** and **GPTMTBILR**. The resulting word ordering for such a write operation is:

```
GPTMTBILR[15:0]:GPTMTAILR[15:0]
```

Likewise, a read access to **GPTMTAR** returns the value:

```
GPTMTBR[15:0]:GPTMTAR[15:0]
```

9.2.2.1 32-Bit One-Shot/Periodic Timer Mode

In 32-bit one-shot and periodic timer modes, the concatenated versions of the TimerA and TimerB registers are configured as a 32-bit down-counter. The selection of one-shot or periodic mode is determined by the value written to the TAMR field of the **GPTM TimerA Mode (GPTMTAMR)** register (see page 170), and there is no need to write to the **GPTM TimerB Mode (GPTMTBMR)** register.

When software writes the TAEN bit in the **GPTM Control (GPTMCTL)** register (see page 174), the timer begins counting down from its preloaded value. Once the 0x0000.0000 state is reached, the timer reloads its start value from the concatenated **GPTMTAILR** on the next cycle. If configured to be a one-shot timer, the timer stops counting and clears the TAEN bit in the **GPTMCTL** register. If configured as a periodic timer, it continues counting.

In addition to reloading the count value, the GPTM generates interrupts and output triggers when it reaches the 0x0000000 state. The GPTM sets the TATORIS bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register (see page 179), and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register (see page 181). If the time-out interrupt is enabled in the **GPTM Interrupt Mask (GPTIMR)** register (see page 177), the GPTM also sets the TATOMIS bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register (see page 180).

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

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

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

9.2.2.2 32-Bit Real-Time Clock Timer Mode

In Real-Time Clock (RTC) mode, the concatenated versions of the TimerA and TimerB registers are configured as a 32-bit up-counter. When RTC mode is selected for the first time, the counter is loaded with a value of 0x0000.0001. All subsequent load values must be written to the **GPTM TimerA Match (GPTMTAMATCHR)** register (see page 185) by the controller.

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

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

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

9.2.3 16-Bit Timer Operating Modes

The GPTM is placed into global 16-bit mode by writing a value of 0x4 to the **GPTM Configuration (GPTMCFG)** register (see page 169). This section describes each of the GPTM 16-bit modes of operation. TimerA and TimerB have identical modes, so a single description is given using an *n* to reference both.

9.2.3.1 16-Bit One-Shot/Periodic Timer Mode

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

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

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

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

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

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

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

Table 9-1. 16-Bit Timer With Prescaler Configurations

Prescale	#Clock (T c)[a]	Max Time	Units
00000000	1	1.3107	mS
00000001	2	2.6214	mS
00000010	3	3.9321	mS
------------	--	--	--
11111100	254	332.9229	mS
11111110	255	334.2336	mS
11111111	256	335.5443	mS

a. Tc is the clock period.

9.2.3.2 16-Bit Input Edge Count Mode

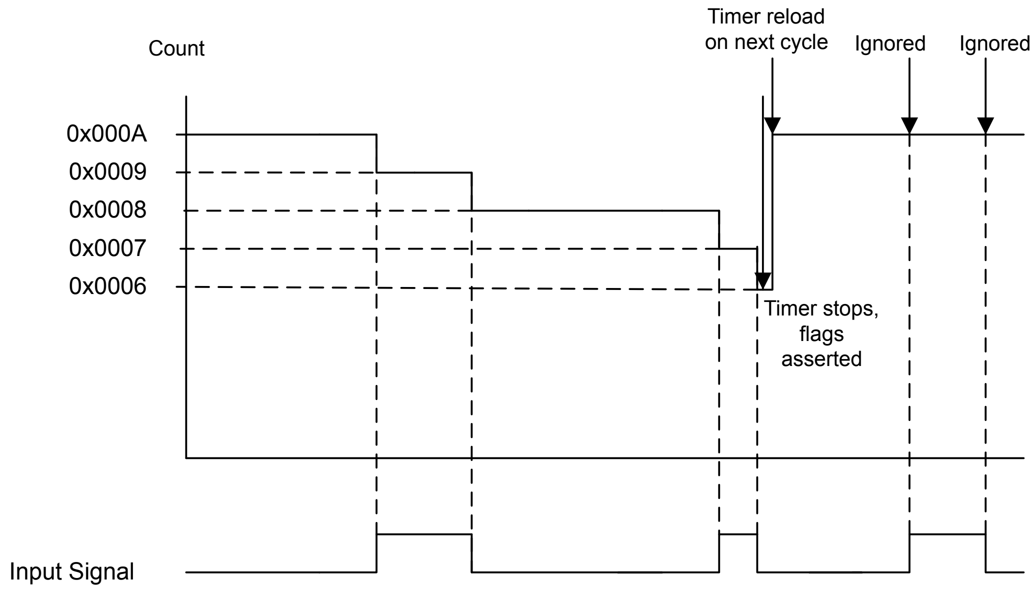
In Edge Count mode, the timer is configured as a down-counter capable of capturing three types of events: rising edge, falling edge, or both. To place the timer in Edge Count mode, the `TnCMR` bit of the **GPTMTnMR** register must be set to 0. The type of edge that the timer counts is determined by the `TnEVENT` fields of the **GPTMCTL** register. During initialization, the **GPTM Timern Match (GPTMTnMATCHR)** register is configured so that the difference between the value in the **GPTMTnILR** register and the **GPTMTnMATCHR** register equals the number of edge events that must be counted.

When software writes the `TnEN` bit in the **GPTM Control (GPTMCTL)** register, the timer is enabled for event capture. Each input event on the `CCP` pin decrements the counter by 1 until the event count matches **GPTMTnMATCHR**. When the counts match, the GPTM asserts the `CnMRIS` bit in the **GPTMRIS** register (and the `CnMMIS` bit, if the interrupt is not masked). The counter is then reloaded using the value in **GPTMTnILR**, and stopped since the GPTM automatically clears the `TnEN` bit in the **GPTMCTL** register. Once the event count has been reached, all further events are ignored until `TnEN` is re-enabled by software.

Figure 9-2 on page 162 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 9-2. 16-Bit Input Edge Count Mode Example



9.2.3.3 16-Bit Input Edge Time Mode

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

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

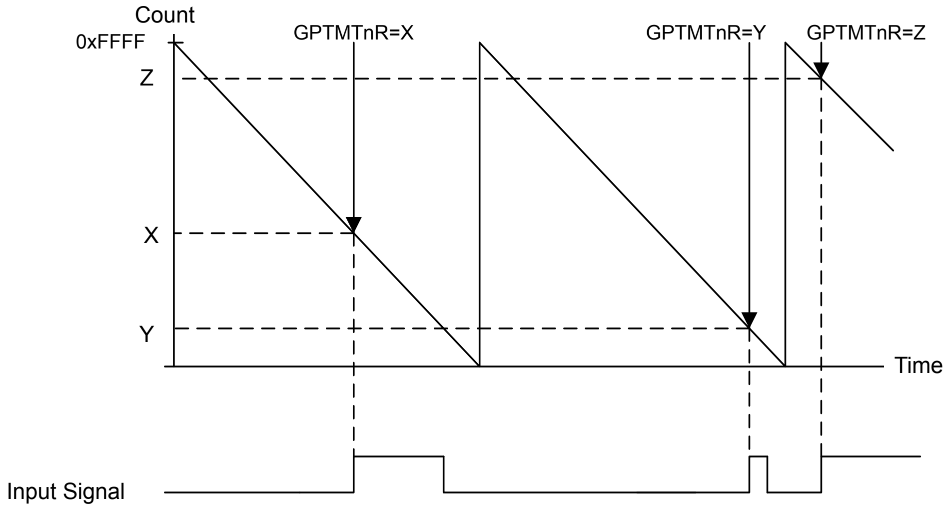
When software writes the `TnEN` bit in the **GPTMCTL** register, the timer is enabled for event capture. When the selected input event is detected, the current **Tn** counter value is captured in the **GPTMTnR** register and is available to be read by the controller. The GPTM then asserts the `CnERIS`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

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

Figure 9-3 on page 163 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 9-3. 16-Bit Input Edge Time Mode Example



9.2.3.4 16-Bit PWM Mode

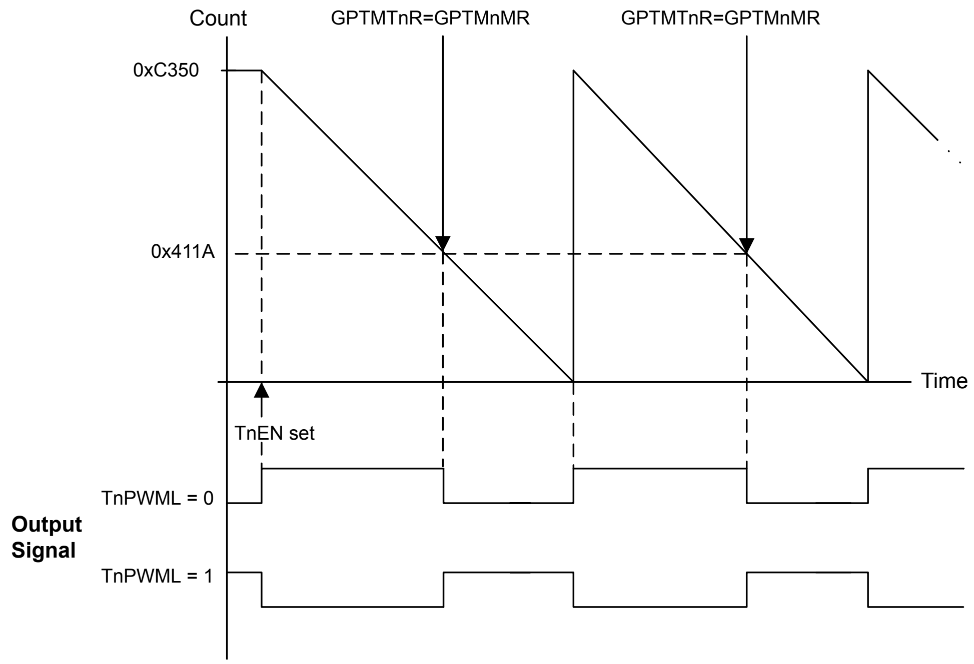
The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a down-counter with a start value (and thus period) defined by **GPTMTnILR**. PWM mode is enabled with the **GPTMTnMR** register by setting the `TnAMS` bit to `0x1`, the `TnCMR` bit to `0x0`, and the `TnMR` field to 0x2.

When software writes the `TnEN` bit in the **GPTMCTL** register, the counter begins counting down until it reaches the 0x0000 state. On the next counter cycle, the counter reloads its start value from **GPTMTnILR** (and **GPTMTnPR** if using a prescaler) and continues counting until disabled by software clearing the `TnEN` bit in the **GPTMCTL** register. No interrupts or status bits are asserted in PWM mode.

The output PWM signal asserts when the counter is at the value of the **GPTMTnILR** register (its start state), and is deasserted when the counter value equals the value in the **GPTM Timern Match Register (GPTMnMATCHR)**. Software has the capability of inverting the output PWM signal by setting the `TnPWML` bit in the **GPTMCTL** register.

Figure 9-4 on page 164 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 9-4. 16-Bit PWM Mode Example



9.3 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.

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

9.3.1 32-Bit One-Shot/Periodic Timer Mode

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

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

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

In One-Shot mode, the timer stops counting after step 7 on page 165. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode does not stop counting after it times out.

9.3.2 32-Bit Real-Time Clock (RTC) Mode

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

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

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

9.3.3 16-Bit One-Shot/Periodic Timer Mode

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

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

In One-Shot mode, the timer stops counting after step 8 on page 165. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode does not stop counting after it times out.

9.3.4 16-Bit Input Edge Count Mode

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

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

In Input Edge Count Mode, the timer stops after the desired number of edge events has been detected. To re-enable the timer, ensure that the `TnEN` bit is cleared and repeat step 4 on page 166-step 9 on page 166.

9.3.5 16-Bit Input Edge Timing Mode

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

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

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

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

9.3.6 16-Bit PWM Mode

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

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

In PWM Timing mode, the timer continues running after the PWM signal has been generated. The PWM period can be adjusted at any time by writing the **GPTMTnILR** register, and the change takes effect at the next cycle after the write.

9.4 Register Map

Table 9-2 on page 167 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

Table 9-2. Timers Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPTMCFG	R/W	0x0000.0000	GPTM Configuration	169
0x004	GPTMTAMR	R/W	0x0000.0000	GPTM TimerA Mode	170
0x008	GPTMTBMR	R/W	0x0000.0000	GPTM TimerB Mode	172

Offset	Name	Type	Reset	Description	See page
0x00C	GPTMCTL	R/W	0x0000.0000	GPTM Control	174
0x018	GPTMIMR	R/W	0x0000.0000	GPTM Interrupt Mask	177
0x01C	GPTMRIS	RO	0x0000.0000	GPTM Raw Interrupt Status	179
0x020	GPTMMIS	RO	0x0000.0000	GPTM Masked Interrupt Status	180
0x024	GPTMICR	W1C	0x0000.0000	GPTM Interrupt Clear	181
0x028	GPTMTAILR	R/W	0x0000.FFFF (16-bit mode) 0xFFFF.FFFF (32-bit mode)	GPTM TimerA Interval Load	183
0x02C	GPTMTBILR	R/W	0x0000.FFFF	GPTM TimerB Interval Load	184
0x030	GPTMTAMATCHR	R/W	0x0000.FFFF (16-bit mode) 0xFFFF.FFFF (32-bit mode)	GPTM TimerA Match	185
0x034	GPTMTBMATCHR	R/W	0x0000.FFFF	GPTM TimerB Match	186
0x038	GPTMTAPR	R/W	0x0000.0000	GPTM TimerA Prescale	187
0x03C	GPTMTBPR	R/W	0x0000.0000	GPTM TimerB Prescale	188
0x040	GPTMTAPMR	R/W	0x0000.0000	GPTM TimerA Prescale Match	189
0x044	GPTMTBPMR	R/W	0x0000.0000	GPTM TimerB Prescale Match	190
0x048	GPTMTAR	RO	0x0000.FFFF (16-bit mode) 0xFFFF.FFFF (32-bit mode)	GPTM TimerA	191
0x04C	GPTMTBR	RO	0x0000.FFFF	GPTM TimerB	192

9.5 Register Descriptions

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

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

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

GPTM Configuration (GPTMCFG)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	GPTMCFG	R/W	0x0	GPTM Configuration The `GPTMCFG` values are defined as follows:

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

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

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

GPTM TimerA Mode (GPTMTAMR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
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												TAAMS	TACMR	TAMR	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
1:0	TAMR	R/W	0x0	GPTM TimerA Mode The TAMR values are defined as follows: 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). In 16-bit timer configuration, TAMR controls the 16-bit timer modes for TimerA. In 32-bit timer configuration, this register controls the mode and the contents of **GPTMTBMR** are ignored.

Register 3: GPTM TimerB Mode (GPTMTBMR), offset 0x008

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

GPTM TimerB Mode (GPTMTBMR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
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												TBAMS	TBCMR	TBMR	
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TBAMS	R/W	0	GPTM TimerB 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 TimerB Capture Mode The `TBCMR` values are defined as follows: Value Description 0 Edge-Count mode. 1 Edge-Time mode.

Bit/Field	Name	Type	Reset	Description
1:0	TBMR	R/W	0x0	GPTM TimerB Mode The TBMR values are defined as follows: 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. In 16-bit timer configuration, these bits control the 16-bit timer modes for TimerB. In 32-bit timer configuration, this register's contents are ignored and **GPTMTAMR** is used.

Register 4: GPTM Control (GPTMCTL), offset 0x00C

This register is used alongside the **GPTMCFG** and **GMTMTnMR** registers to fine-tune the timer configuration, and to enable other features such as timer stall and the output trigger.

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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	TBPWML	R/W	0	GPTM TimerB PWM Output Level The `TBPWML` values are defined as follows: Value Description 0 Output is unaffected. 1 Output is inverted.
13	TBOTE	R/W	0	GPTM TimerB Output Trigger Enable The `TBOTE` values are defined as follows: Value Description 0 The output TimerB trigger is disabled. 1 The output TimerB trigger is enabled.
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11:10	TBEVENT	R/W	0x0	GPTM TimerB Event Mode The `TBEVENT` values are defined as follows: Value Description 0x0 Positive edge. 0x1 Negative edge. 0x2 Reserved 0x3 Both edges.

Bit/Field	Name	Type	Reset	Description
9	TBSTALL	R/W	0	GPTM TimerB Stall Enable The TBSTALL values are defined as follows: Value Description 0 TimerB stalling is disabled. 1 TimerB stalling is enabled.
8	TBEN	R/W	0	GPTM TimerB Enable The TBEN values are defined as follows: Value Description 0 TimerB is disabled. 1 TimerB is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	TAPWML	R/W	0	GPTM TimerA PWM Output Level The TAPWML values are defined as follows: Value Description 0 Output is unaffected. 1 Output is inverted.
5	TAOTE	R/W	0	GPTM TimerA Output Trigger Enable The TAOTE values are defined as follows: Value Description 0 The output TimerA trigger is disabled. 1 The output TimerA trigger is enabled.
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.

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

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

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

GPTM Interrupt Mask (GPTMIMR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
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					CBEIM	CBMIM	TBTOIM	reserved				RTCIM	CAEIM	CAMIM	TATOIM
Type	RO	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Bit/Field	Name	Type	Reset	Description
3	RTCIM	R/W	0	GPTM RTC Interrupt Mask The `RTCIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
2	CAEIM	R/W	0	GPTM CaptureA Event Interrupt Mask The `CAEIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
1	CAMIM	R/W	0	GPTM CaptureA Match Interrupt Mask The `CAMIM` values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
0	TATOIM	R/W	0	GPTM TimerA Time-Out Interrupt Mask The `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					CBERIS	CBMRIS	TBTORIS	reserved				RTCRIS	CAERIS	CAMRIS	TATORIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Register 7: GPTM Masked Interrupt Status (GPTMMIS), offset 0x020

This register show the state of the GPTM's controller-level interrupt. If an interrupt is unmasked in **GPTMIMR**, and there is an event that causes the interrupt to be asserted, the corresponding bit is set in this register. All bits are cleared by writing a 1 to the corresponding bit in **GPTMICR**.

GPTM Masked Interrupt Status (GPTMMIS)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
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					CBEMIS	CBMMIS	TBTOMIS	reserved				RTCMIS	CAEMIS	CAMMIS	TATOMIS
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBEMIS	RO	0	GPTM CaptureB Event Masked Interrupt This is the CaptureB event interrupt status after masking.
9	CBMMIS	RO	0	GPTM CaptureB Match Masked Interrupt This is the CaptureB match interrupt status after masking.
8	TBTOMIS	RO	0	GPTM TimerB Time-Out Masked Interrupt This is the TimerB time-out interrupt status after masking.
7:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	RTCMIS	RO	0	GPTM RTC Masked Interrupt This is the RTC event interrupt status after masking.
2	CAEMIS	RO	0	GPTM CaptureA Event Masked Interrupt This is the CaptureA event interrupt status after masking.
1	CAMMIS	RO	0	GPTM CaptureA Match Masked Interrupt This is the CaptureA match interrupt status after masking.
0	TATOMIS	RO	0	GPTM TimerA Time-Out Masked Interrupt This is the TimerA time-out interrupt status after masking.

Register 8: GPTM Interrupt Clear (GPTMICR), offset 0x024

This register is used to clear the status bits in the **GPTMRIS** and **GPTMMIS** registers. Writing a 1 to a bit clears the corresponding bit in the **GPTMRIS** and **GPTMMIS** registers.

GPTM Interrupt Clear (GPTMICR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
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					CBECINT	CBMCINT	TBTOCINT	reserved				RTCCINT	CAECINT	CAMCINT	TATOCINT
Type	RO	RO	RO	RO	RO	W1C	W1C	W1C	RO	RO	RO	RO	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBECINT	W1C	0	GPTM CaptureB Event Interrupt Clear The `CBECINT` values are defined as follows: Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
9	CBMCINT	W1C	0	GPTM CaptureB Match Interrupt Clear The `CBMCINT` values are defined as follows: Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
8	TBTOCINT	W1C	0	GPTM TimerB Time-Out Interrupt Clear The `TBTOCINT` values are defined as follows: Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
7:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
3	RTCCINT	W1C	0	GPTM RTC Interrupt Clear The `RTCCINT` values are defined as follows: Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
2	CAECINT	W1C	0	GPTM CaptureA Event Interrupt Clear The `CAECINT` values are defined as follows: Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
1	CAMCINT	W1C	0	GPTM CaptureA Match Raw Interrupt This is the CaptureA match interrupt status after masking.
0	TATOCINT	W1C	0	GPTM TimerA Time-Out Raw Interrupt The `TATOCINT` values are defined as follows: Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.

Register 9: GPTM TimerA Interval Load (GPTMTAILR), offset 0x028

This register is used to load the starting count value into the timer. When GPTM is configured to one of the 32-bit modes, **GPTMTAILR** appears as a 32-bit register (the upper 16-bits correspond to the contents of the **GPTM TimerB Interval Load (GPTMTBILR)** register). In 16-bit mode, the upper 16 bits of this register read as 0s and have no effect on the state of **GPTMTBILR**.

GPTM TimerA Interval Load (GPTMTAILR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x028
Type R/W, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)

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

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

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

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

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

GPTM TimerB Interval Load (GPTMTBILR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
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 TimerB Interval Load Register When the GPTM is not configured as a 32-bit timer, a write to this field updates **GPTMTBILR**. In 32-bit mode, writes are ignored, and reads return the current value of **GPTMTBILR**.

Register 11: GPTM TimerA Match (GPTMTAMATCHR), offset 0x030

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

GPTM TimerA Match (GPTMTAMATCHR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x030
Type R/W, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)

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

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

Bit/Field	Name	Type	Reset	Description
31:16	TAMRH	R/W	0xFFFF (32-bit mode) 0x0000 (16-bit mode)	GPTM TimerA Match Register High When configured for 32-bit Real-Time Clock (RTC) mode via the **GPTMCFG** register, this value is compared to the upper half of **GPTMTAR**, to determine match events. 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 TimerA Match Register Low When configured for 32-bit Real-Time Clock (RTC) mode via the **GPTMCFG** register, this value is compared to the lower half of **GPTMTAR**, to determine match events. When configured for PWM mode, this value along with **GPTMTAILR**, determines the duty cycle of the output PWM signal. When configured for Edge Count mode, this value along with **GPTMTAILR**, determines how many edge events are counted. The total number of edge events counted is equal to the value in **GPTMTAILR** minus this value.

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

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

GPTM TimerB Match (GPTMTBMATCHR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
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 TimerB Match Register Low When configured for PWM mode, this value along with **GPTMTBILR**, determines the duty cycle of the output PWM signal. When configured for Edge Count mode, this value along with **GPTMTBILR**, determines how many edge events are counted. The total number of edge events counted is equal to the value in **GPTMTBILR** minus this value.

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

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

GPTM TimerA Prescale (GPTMTAPR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
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	0x00	Software should not rely on the value of a reserved 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 TimerA Prescale The register loads this value on a write. A read returns the current value of the register. Refer to Table 9-1 on page 161 for more details and an example.

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

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

GPTM TimerB Prescale (GPTMTBPR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x03C
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TBPSR	R/W	0x00	GPTM TimerB Prescale The register loads this value on a write. A read returns the current value of this register. Refer to Table 9-1 on page 161 for more details and an example.

Register 15: GPTM TimerA Prescale Match (GPTMTAPMR), offset 0x040

This register effectively extends the range of **GPTMTAMATCHR** to 24 bits when operating in 16-bit one-shot or periodic mode.

GPTM TimerA Prescale Match (GPTMTAPMR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x040
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TAPSMR	R/W	0x00	GPTM TimerA Prescale Match This value is used alongside **GPTMTAMATCHR** to detect timer match events while using a prescaler.

Register 16: GPTM TimerB Prescale Match (GPTMTBPMR), offset 0x044

This register effectively extends the range of **GPTMTBMATCHR** to 24 bits when operating in 16-bit one-shot or periodic mode.

GPTM TimerB Prescale Match (GPTMTBPMR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x044
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								TBPSMR							
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	TBPSMR	R/W	0x00	GPTM TimerB Prescale Match This value is used alongside **GPTMTBMATCHR** to detect timer match events while using a prescaler.

Register 17: GPTM TimerA (GPTMTAR), offset 0x048

This register shows the current value of the TimerA counter in all cases except for Input Edge Count mode. When in this mode, this register contains the time at which the last edge event took place.

GPTM TimerA (GPTMTAR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Offset 0x048
Type RO, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)

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

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

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

Register 18: GPTM TimerB (GPTMTBR), offset 0x04C

This register shows the current value of the TimerB counter in all cases except for Input Edge Count mode. When in this mode, this register contains the time at which the last edge event took place.

GPTM TimerB (GPTMTBR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
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 TimerB A read returns the current value of the **GPTM TimerB Count Register**, except in Input Edge Count mode, when it returns the timestamp from the last edge event.

10 Watchdog Timer

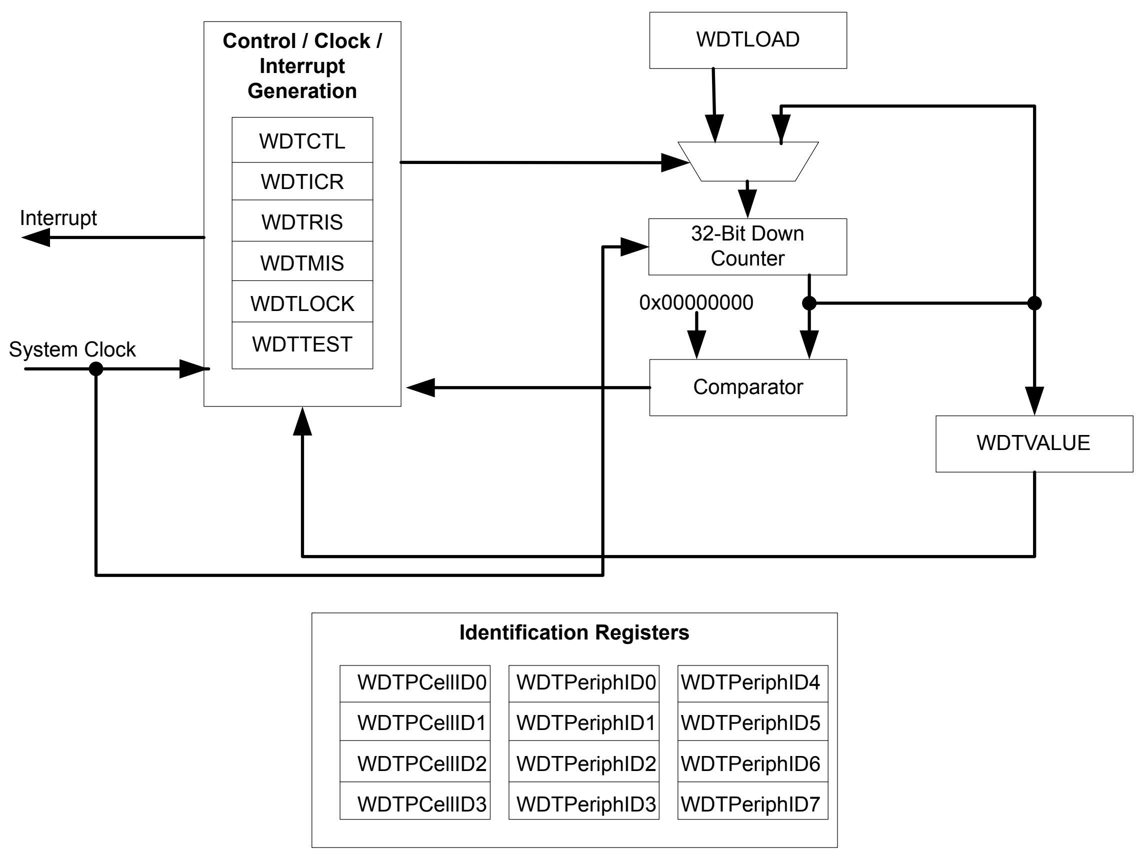
A watchdog timer can generate nonmaskable interrupts (NMIs) or a reset when a time-out value is reached. The watchdog timer is used to regain control when a system has failed due to a software error or due to the failure of an external device to respond in the expected way.

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

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

10.1 Block Diagram

Figure 10-1. WDT Module Block Diagram



10.2 Functional Description

The Watchdog Timer module generates the first time-out signal when the 32-bit counter reaches the zero state after being enabled; enabling the counter also enables the watchdog timer interrupt. After the first time-out event, the 32-bit counter is re-loaded with the value of the **Watchdog Timer Load (WDTLOAD)** register, and the timer resumes counting down from that value. Once the

Watchdog Timer has been configured, the **Watchdog Timer Lock (WDTLOCK)** register is written, which prevents the timer configuration from being inadvertently altered by software.

If the timer counts down to its zero state again before the first time-out interrupt is cleared, and the reset signal has been enabled (via the `WatchdogResetEnable` function), the Watchdog timer asserts its reset signal to the system. If the interrupt is cleared before the 32-bit counter reaches its second time-out, the 32-bit counter is loaded with the value in the **WDTLOAD** register, and counting resumes from that value.

If **WDTLOAD** is written with a new value while the Watchdog Timer counter is counting, then the counter is loaded with the new value and continues counting.

Writing to **WDTLOAD** does not clear an active interrupt. An interrupt must be specifically cleared by writing to the **Watchdog Interrupt Clear (WDTICR)** register.

The Watchdog module interrupt and reset generation can be enabled or disabled as required. When the interrupt is re-enabled, the 32-bit counter is preloaded with the load register value and not its last state.

10.3 Initialization and Configuration

To use the WDT, its peripheral clock must be enabled by setting the `WDT` bit in the **RCGC0** register. The Watchdog Timer is configured using the following sequence:

1. Load the **WDTLOAD** register with the desired timer load value.

2. If the Watchdog is configured to trigger system resets, set the `RESEN` bit in the **WDTCTL** register.

3. Set the `INTEN` bit in the **WDTCTL** register to enable the Watchdog and lock the control register.

If software requires that all of the watchdog registers are locked, the Watchdog Timer module can be fully locked by writing any value to the **WDTLOCK** register. To unlock the Watchdog Timer, write a value of 0x1ACC.E551.

10.4 Register Map

Table 10-1 on page 194 lists the Watchdog registers. The offset listed is a hexadecimal increment to the register's address, relative to the Watchdog Timer base address of 0x4000.0000.

Table 10-1. Watchdog Timer Register Map

Offset	Name	Type	Reset	Description	See page
0x000	WDTLOAD	R/W	0xFFFF.FFFF	Watchdog Load	196
0x004	WDTVALUE	RO	0xFFFF.FFFF	Watchdog Value	197
0x008	WDTCTL	R/W	0x0000.0000	Watchdog Control	198
0x00C	WDTICR	WO	-	Watchdog Interrupt Clear	199
0x010	WDTRIS	RO	0x0000.0000	Watchdog Raw Interrupt Status	200
0x014	WDTMIS	RO	0x0000.0000	Watchdog Masked Interrupt Status	201
0x418	WDTTEST	R/W	0x0000.0000	Watchdog Test	202
0xC00	WDTLOCK	R/W	0x0000.0000	Watchdog Lock	203

Offset	Name	Type	Reset	Description	See page
0xFD0	WDTPeriphID4	RO	0x0000.0000	Watchdog Peripheral Identification 4	204
0xFD4	WDTPeriphID5	RO	0x0000.0000	Watchdog Peripheral Identification 5	205
0xFD8	WDTPeriphID6	RO	0x0000.0000	Watchdog Peripheral Identification 6	206
0xFDC	WDTPeriphID7	RO	0x0000.0000	Watchdog Peripheral Identification 7	207
0xFE0	WDTPeriphID0	RO	0x0000.0005	Watchdog Peripheral Identification 0	208
0xFE4	WDTPeriphID1	RO	0x0000.0018	Watchdog Peripheral Identification 1	209
0xFE8	WDTPeriphID2	RO	0x0000.0018	Watchdog Peripheral Identification 2	210
0xFEC	WDTPeriphID3	RO	0x0000.0001	Watchdog Peripheral Identification 3	211
0xFF0	WDTPCellID0	RO	0x0000.000D	Watchdog PrimeCell Identification 0	212
0xFF4	WDTPCellID1	RO	0x0000.00F0	Watchdog PrimeCell Identification 1	213
0xFF8	WDTPCellID2	RO	0x0000.0005	Watchdog PrimeCell Identification 2	214
0xFFC	WDTPCellID3	RO	0x0000.00B1	Watchdog PrimeCell Identification 3	215

10.5 Register Descriptions

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

Register 1: Watchdog Load (WDTLOAD), offset 0x000

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

Watchdog Load (WDTLOAD)

Base 0x4000.0000
Offset 0x000
Type R/W, reset 0xFFFF.FFFF

	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)

Base 0x4000.0000
Offset 0x004
Type RO, reset 0xFFFF.FFFF

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

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

Bit/Field	Name	Type	Reset	Description
31:0	WDTValue	RO	0xFFFF.FFFF	Watchdog Value Current value of the 32-bit down counter.

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

This register is the watchdog control register. The watchdog timer can be configured to generate a reset signal (on second time-out) or an interrupt on time-out.

When the watchdog interrupt has been enabled, all subsequent writes to the control register are ignored. The only mechanism that can re-enable writes is a hardware reset.

Watchdog Control (WDTCTL)

Base 0x4000.0000
Offset 0x008
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														RESEN	INTEN
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W
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	RESEN	R/W	0	Watchdog Reset Enable The `RESEN` values are defined as follows: Value Description 0 Disabled. 1 Enable the Watchdog module reset output.
0	INTEN	R/W	0	Watchdog Interrupt Enable The `INTEN` values are defined as follows: Value Description 0 Interrupt event disabled (once this bit is set, it can only be cleared by a hardware reset). 1 Interrupt event enabled. Once enabled, all writes are ignored.

Register 4: Watchdog Interrupt Clear (WDTICR), offset 0x00C

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

Watchdog Interrupt Clear (WDTICR)

Base 0x4000.0000
Offset 0x00C
Type WO, reset -

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	WDTIntClr															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WDTIntClr															
Type	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO	WO
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Bit/Field	Name	Type	Reset	Description
31:0	WDTIntClr	WO	-	Watchdog Interrupt Clear

Register 5: Watchdog Raw Interrupt Status (WDTRIS), offset 0x010

This register is the raw interrupt status register. Watchdog interrupt events can be monitored via this register if the controller interrupt is masked.

Watchdog Raw Interrupt Status (WDTRIS)

Base 0x4000.0000
Offset 0x010
Type RO, reset 0x0000.0000

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

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

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

Register 6: Watchdog Masked Interrupt Status (WDTMIS), offset 0x014

This register is the masked interrupt status register. The value of this register is the logical AND of the raw interrupt bit and the Watchdog interrupt enable bit.

Watchdog Masked Interrupt Status (WDTMIS)

Base 0x4000.0000
Offset 0x014
Type RO, reset 0x0000.0000

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

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

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

Register 7: Watchdog Test (WDTTEST), offset 0x418

This register provides user-enabled stalling when the microcontroller asserts the CPU halt flag during debug.

Watchdog Test (WDTTEST)

Base 0x4000.0000
Offset 0x418
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	STALL	R/W	0	Watchdog Stall Enable When set to 1, if the Stellaris® microcontroller is stopped with a debugger, the watchdog timer stops counting. Once the microcontroller is restarted, the watchdog timer resumes counting.
7:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 8: Watchdog Lock (WDTLOCK), offset 0xC00

Writing 0x1ACC.E551 to the **WDTLOCK** register enables write access to all other registers. Writing any other value to the **WDTLOCK** register re-enables the locked state for register writes to all the other registers. Reading the **WDTLOCK** register returns the lock status rather than the 32-bit value written. Therefore, when write accesses are disabled, reading the **WDTLOCK** register returns 0x0000.0001 (when locked; otherwise, the returned value is 0x0000.0000 (unlocked)).

Watchdog Lock (WDTLOCK)

Base 0x4000.0000
Offset 0xC00
Type R/W, reset 0x0000.0000

	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	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)

Base 0x4000.0000
Offset 0xFD0
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Register 10: Watchdog Peripheral Identification 5 (WDTPeriphID5), offset 0xFD4

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

Watchdog Peripheral Identification 5 (WDTPeriphID5)

Base 0x4000.0000
Offset 0xFD4
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID5							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Register 11: Watchdog Peripheral Identification 6 (WDTPeriphID6), offset 0xFD8

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

Watchdog Peripheral Identification 6 (WDTPeriphID6)

Base 0x4000.0000
Offset 0xFD8
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID6							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Register 12: Watchdog Peripheral Identification 7 (WDTPeriphID7), offset 0xFDC

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

Watchdog Peripheral Identification 7 (WDTPeriphID7)

Base 0x4000.0000
Offset 0xFDC
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID7							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Register 13: Watchdog Peripheral Identification 0 (WDTPeriphID0), offset 0xFE0

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

Watchdog Peripheral Identification 0 (WDTPeriphID0)

Base 0x4000.0000
Offset 0xFE0
Type RO, reset 0x0000.0005

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

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

Register 14: Watchdog Peripheral Identification 1 (WDTPeriphID1), offset 0xFE4

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

Watchdog Peripheral Identification 1 (WDTPeriphID1)

Base 0x4000.0000
Offset 0xFE4
Type RO, reset 0x0000.0018

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x18	Watchdog Peripheral ID Register[15:8]

Register 15: Watchdog Peripheral Identification 2 (WDTPeriphID2), offset 0xFE8

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

Watchdog Peripheral Identification 2 (WDTPeriphID2)

Base 0x4000.0000
Offset 0xFE8
Type RO, reset 0x0000.0018

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	Watchdog Peripheral ID Register[23:16]

Register 16: Watchdog Peripheral Identification 3 (WDTPeriphID3), offset 0xFEC

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

Watchdog Peripheral Identification 3 (WDTPeriphID3)

Base 0x4000.0000
Offset 0xFEC
Type RO, reset 0x0000.0001

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

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

Register 17: Watchdog PrimeCell Identification 0 (WDTPCellID0), offset 0xFF0

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

Watchdog PrimeCell Identification 0 (WDTPCellID0)

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

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

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

Register 18: Watchdog PrimeCell Identification 1 (WDTPCellID1), offset 0xFF4

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

Watchdog PrimeCell Identification 1 (WDTPCellID1)

Base 0x4000.0000
Offset 0xFF4
Type RO, reset 0x0000.00F0

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

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

Register 19: Watchdog PrimeCell Identification 2 (WDTPCellID2), offset 0xFF8

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

Watchdog PrimeCell Identification 2 (WDTPCellID2)

Base 0x4000.0000
Offset 0xFF8
Type RO, reset 0x0000.0005

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	Watchdog PrimeCell ID Register[23:16]

Register 20: Watchdog PrimeCell Identification 3 (WDTPCellID3), offset 0xFFC

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

Watchdog PrimeCell Identification 3 (WDTPCellID3)

Base 0x4000.0000
Offset 0xFFC
Type RO, reset 0x0000.00B1

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved 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]

11 Universal Asynchronous Receivers/Transmitters (UARTs)

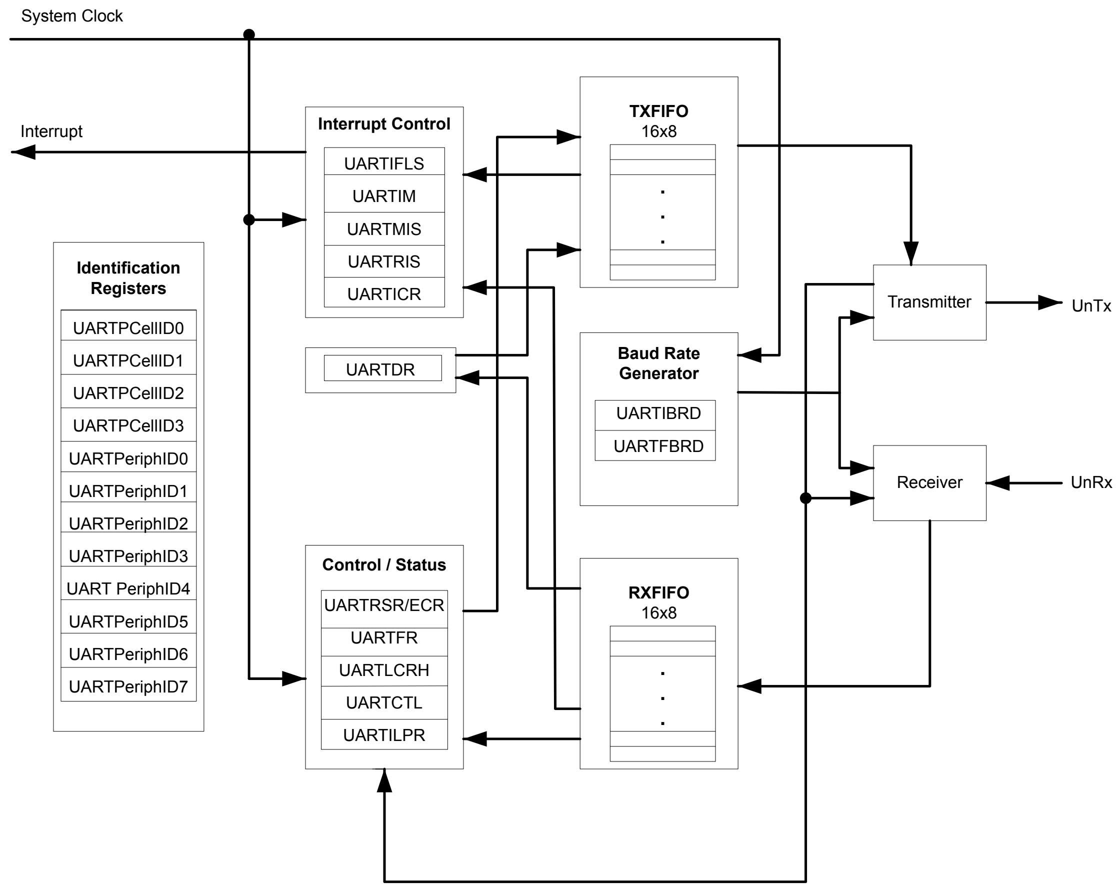
The Stellaris® Universal Asynchronous Receiver/Transmitter (UART) provides fully programmable, 16C550-type serial interface characteristics. The LM3S600 controller is equipped with two UART modules.

Each UART has the following features:

- Separate transmit and receive FIFOs
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
- Programmable baud-rate generator allowing rates up to 3.125 Mbps
- Standard asynchronous communication bits for start, stop, and parity
- False start bit detection
- Line-break generation and detection
- Fully programmable serial interface characteristics:
  - 5, 6, 7, or 8 data bits
  - Even, odd, stick, or no-parity bit generation/detection
  - 1 or 2 stop bit generation

11.1 Block Diagram

Figure 11-1. UART Module Block Diagram



11.2 Functional Description

Each Stellaris® UART performs the functions of parallel-to-serial and serial-to-parallel conversions. It is similar in functionality to a 16C550 UART, but is not register compatible.

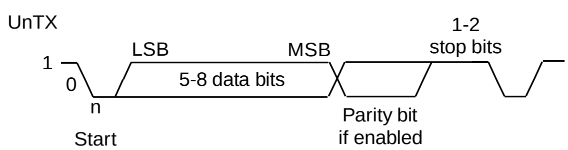
The UART is configured for transmit and/or receive via the TXE and RXE bits of the **UART Control (UARTCTL)** register (see page 233). Transmit and receive are both enabled out of reset. Before any control registers are programmed, the UART must be disabled by clearing the UARTEN bit in **UARTCTL**. If the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

11.2.1 Transmit/Receive Logic

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. The control logic outputs the serial bit stream beginning with a start bit, and followed by the data bits (LSB first), parity bit, and the stop bits according to the programmed configuration in the control registers. See Figure 11-2 on page 218 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 11-2. UART Character Frame



11.2.2 Baud-Rate Generation

The baud-rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. The number formed by these two values is used by the baud-rate generator to determine the bit period. Having a fractional baud-rate divider allows the UART to generate all the standard baud rates.

The 16-bit integer is loaded through the **UART Integer Baud-Rate Divisor (UARTIBRD)** register (see page 229) and the 6-bit fractional part is loaded with the **UART Fractional Baud-Rate Divisor (UARTFBRD)** register (see page 230). 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 = SysClk / (16 * Baud Rate)
```

The 6-bit fractional number (that is to be loaded into the `DIVFRAC` bit field in the **UARTFBRD** register) can be calculated by taking the fractional part of the baud-rate divisor, multiplying it by 64, and adding 0.5 to account for rounding errors:

```
UARTFBRD[DIVFRAC] = integer(BRDF * 64 + 0.5)
```

The UART generates an internal baud-rate reference clock at 16x the baud-rate (referred to as `Baud16`). This reference clock is divided by 16 to generate the transmit clock, and is used for error detection during receive operations.

Along with the **UART Line Control, High Byte (UARTLCRH)** register (see page 231), 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

11.2.3 Data Transmission

Data received or transmitted is stored in two 16-byte FIFOs, though the receive FIFO has an extra four bits per character for status information. For transmission, data is written into the transmit FIFO. If the UART is enabled, it causes a data frame to start transmitting with the parameters indicated in the **UARTLCRH** register. Data continues to be transmitted until there is no data left in the transmit FIFO. The BUSY bit in the **UART Flag (UARTFR)** register (see page 227) is asserted as soon as data is written to the transmit FIFO (that is, if the FIFO is non-empty) and remains asserted while data is being transmitted. The BUSY bit is negated only when the transmit FIFO is empty, and the last character has been transmitted from the shift register, including the stop bits. The UART can indicate that it is busy even though the UART may no longer be enabled.

When the receiver is idle (the UnRx is continuously 1) and the data input goes Low (a start bit has been received), the receive counter begins running and data is sampled on the eighth cycle of Baud16 (described in “Transmit/Receive Logic” on page 217).

The start bit is valid if UnRx is still low on the eighth cycle of Baud16, otherwise a false start bit is detected and it is ignored. Start bit errors can be viewed in the **UART Receive Status (UARTRSR)** register (see page 225). If the start bit was valid, successive data bits are sampled on every 16th cycle of Baud16 (that is, one bit period later) according to the programmed length of the data characters. The parity bit is then checked if parity mode was enabled. Data length and parity are defined in the **UARTLCRH** register.

Lastly, a valid stop bit is confirmed if UnRx is High, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO, with any error bits associated with that word.

11.2.4 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 223). 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 231).

FIFO status can be monitored via the **UART Flag (UARTFR)** register (see page 227) 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 234). Both FIFOs can be individually configured to trigger interrupts at different levels. Available configurations include 1/8, ¼, ½, ¾, and 7/8. 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.

11.2.5 Interrupts

The UART can generate interrupts when the following conditions are observed:

- Overrun Error
- Break Error

- Parity Error
- Framing Error
- Receive Timeout
- Transmit (when condition defined in the `TXIFLSEL` bit in the **UARTIFLS** register is met)
- Receive (when condition defined in the `RXIFLSEL` bit in the **UARTIFLS** register is met)

All of the interrupt events are ORed together before being sent to the interrupt controller, so the UART can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine by reading the **UART Masked Interrupt Status (UARTMIS)** register (see page 239).

The interrupt events that can trigger a controller-level interrupt are defined in the **UART Interrupt Mask (UARTIM**) register (see page 236) by setting the corresponding `IM` bit to 1. If interrupts are not used, the raw interrupt status is always visible via the **UART Raw Interrupt Status (UARTRIS)** register (see page 238).

Interrupts are always cleared (for both the **UARTMIS** and **UARTRIS** registers) by setting the corresponding bit in the **UART Interrupt Clear (UARTICR)** register (see page 240).

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.

11.2.6 Loopback Operation

The UART can be placed into an internal loopback mode for diagnostic or debug work. This is accomplished by setting the `LBE` bit in the **UARTCTL** register (see page 233). In loopback mode, data transmitted on UnTx is received on the UnRx input.

11.3 Initialization and Configuration

To use the UARTs, the peripheral clock must be enabled by setting the `UART0` or `UART1` bits in the **RCGC1** register.

This section discusses the steps that are required for using a UART module. For this example, the system clock is assumed to be 20 MHz and the desired UART configuration is:

- 115200 baud rate
- Data length of 8 bits
- One stop bit
- No parity
- FIFOs disabled
- No interrupts

The first thing to consider when programming the UART is the baud-rate divisor (BRD), since the **UARTIBRD** and **UARTFBRD** registers must be written before the **UARTLCRH** register. Using the equation described in “Baud-Rate Generation” on page 218, 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 229) should be set to 10. The value to be loaded into the **UARTFBRD** register (see page 230) 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. Enable the UART by setting the UARTEN bit in the **UARTCTL** register.

11.4 Register Map

Table 11-1 on page 221 lists the UART registers. The offset listed is a hexadecimal increment to the register's address, relative to that UART's base address:

- UART0: 0x4000.C000
- UART1: 0x4000.D000

Note: The UART must be disabled (see the UARTEN bit in the **UARTCTL** register on page 233) 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 11-1. UART Register Map

Offset	Name	Type	Reset	Description	See page
0x000	UARTDR	R/W	0x0000.0000	UART Data	223
0x004	UARTRSR/UARTECR	R/W	0x0000.0000	UART Receive Status/Error Clear	225
0x018	UARTFR	RO	0x0000.0090	UART Flag	227
0x024	UARTIBRD	R/W	0x0000.0000	UART Integer Baud-Rate Divisor	229
0x028	UARTFBRD	R/W	0x0000.0000	UART Fractional Baud-Rate Divisor	230
0x02C	UARTLCRH	R/W	0x0000.0000	UART Line Control	231
0x030	UARTCTL	R/W	0x0000.0300	UART Control	233
0x034	UARTIFLS	R/W	0x0000.0012	UART Interrupt FIFO Level Select	234
0x038	UARTIM	R/W	0x0000.0000	UART Interrupt Mask	236
0x03C	UARTRIS	RO	0x0000.000F	UART Raw Interrupt Status	238
0x040	UARTMIS	RO	0x0000.0000	UART Masked Interrupt Status	239

Offset	Name	Type	Reset	Description	See page
0x044	UARTICR	W1C	0x0000.0000	UART Interrupt Clear	240
0xFD0	UARTPeriphID4	RO	0x0000.0000	UART Peripheral Identification 4	242
0xFD4	UARTPeriphID5	RO	0x0000.0000	UART Peripheral Identification 5	243
0xFD8	UARTPeriphID6	RO	0x0000.0000	UART Peripheral Identification 6	244
0xFDC	UARTPeriphID7	RO	0x0000.0000	UART Peripheral Identification 7	245
0xFE0	UARTPeriphID0	RO	0x0000.0011	UART Peripheral Identification 0	246
0xFE4	UARTPeriphID1	RO	0x0000.0000	UART Peripheral Identification 1	247
0xFE8	UARTPeriphID2	RO	0x0000.0018	UART Peripheral Identification 2	248
0xFEC	UARTPeriphID3	RO	0x0000.0001	UART Peripheral Identification 3	249
0xFF0	UARTPCellID0	RO	0x0000.000D	UART PrimeCell Identification 0	250
0xFF4	UARTPCellID1	RO	0x0000.00F0	UART PrimeCell Identification 1	251
0xFF8	UARTPCellID2	RO	0x0000.0005	UART PrimeCell Identification 2	252
0xFFC	UARTPCellID3	RO	0x0000.00B1	UART PrimeCell Identification 3	253

11.5 Register Descriptions

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

Register 1: UART Data (UARTDR), offset 0x000

This register is the data register (the interface to the FIFOs).

When FIFOs are enabled, data written to this location is pushed onto the transmit FIFO. If FIFOs are disabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). A write to this register initiates a transmission from the UART.

For received data, if the FIFO is enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO. If FIFOs are disabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO). The received data can be retrieved by reading this register.

UART Data (UARTDR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x000
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved				OE	BE	PE	FE	DATA							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	OE	RO	0	UART Overrun Error The OE values are defined as follows: Value Description 0 There has been no data loss due to a FIFO overrun. 1 New data was received when the FIFO was full, resulting in data loss.
10	BE	RO	0	UART Break Error This bit is set to 1 when a break condition is detected, indicating that the receive data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits). In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the received data input goes to a 1 (marking state) and the next valid start bit is received.

Bit/Field	Name	Type	Reset	Description
9	PE	RO	0	UART Parity Error This bit is set to 1 when the parity of the received data character does not match the parity defined by bits 2 and 7 of the **UARTLCRH** register. In FIFO mode, this error is associated with the character at the top of the FIFO.
8	FE	RO	0	UART Framing Error This bit is set to 1 when the received character does not have a valid stop bit (a valid stop bit is 1).
7:0	DATA	R/W	0	Data Transmitted or Received When written, the data that is to be transmitted via the UART. When read, 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.

A write of any value to the **UARTECR** register clears the framing, parity, break, and overrun errors. All the bits are cleared to 0 on reset.

Read-Only Receive Status (UARTRSR) Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x004
Type RO, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation. The **UARTRSR** register cannot be written.
3	OE	RO	0	UART Overrun Error When this bit is set to 1, data is received and the FIFO is already full. This bit is cleared to 0 by a write to **UARTECR**. The FIFO contents remain valid since no further data is written when the FIFO is full, only the contents of the shift register are overwritten. The CPU must now read the data in order to empty the FIFO.
2	BE	RO	0	UART Break Error This bit is set to 1 when a break condition is detected, indicating that the received data input was held Low for longer than a full-word transmission time (defined as start, data, parity, and stop bits). This bit is cleared to 0 by a write to **UARTECR**. In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state) and the next valid start bit is received.

Bit/Field	Name	Type	Reset	Description
1	PE	RO	0	UART Parity Error This bit is set to 1 when the parity of the received data character does not match the parity defined by bits 2 and 7 of the **UARTLCRH** register. This bit is cleared to 0 by a write to **UARTECR**.
0	FE	RO	0	UART Framing Error This bit is set to 1 when the received character does not have a valid stop bit (a valid stop bit is 1). 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 (UARTECR) Register

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	WO	0	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
Offset 0x018
Type RO, reset 0x0000.0090

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								TXFE	RXFF	TXFF	RXFE	BUSY	reserved		
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TXFE	RO	1	UART Transmit FIFO Empty The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. If the FIFO is disabled (FEN is 0), this bit is set when the transmit holding register is empty. If the FIFO is enabled (FEN is 1), this bit is set when the transmit FIFO is empty.
6	RXFF	RO	0	UART Receive FIFO Full The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. If the FIFO is disabled, this bit is set when the receive holding register is full. If the FIFO is enabled, this bit is set when the receive FIFO is full.
5	TXFF	RO	0	UART Transmit FIFO Full The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. If the FIFO is disabled, this bit is set when the transmit holding register is full. If the FIFO is enabled, this bit is set when the transmit FIFO is full.

Bit/Field	Name	Type	Reset	Description
4	RXFE	RO	1	UART Receive FIFO Empty The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. If the FIFO is disabled, this bit is set when the receive holding register is empty. If the FIFO is enabled, this bit is set when the receive FIFO is empty.
3	BUSY	RO	0	UART Busy When this bit is 1, the UART is busy transmitting data. This bit remains set until the complete byte, including all stop bits, has been sent from the shift register. 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 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 218 for configuration details.

UART Integer Baud-Rate Divisor (UARTIBRD)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x024
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	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	0	Software should not rely on the value of 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 5: 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 218 for configuration details.

UART Fractional Baud-Rate Divisor (UARTFBRD)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x028
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	DIVFRAC	R/W	0x000	Fractional Baud-Rate Divisor

Register 6: UART Line Control (UARTLCRH), offset 0x02C

The **UARTLCRH** register is the line control register. Serial parameters such as data length, parity, and stop bit selection are implemented in this register.

When updating the baud-rate divisor (**UARTIBRD** and/or **UARTIFRD**), the **UARTLCRH** register must also be written. The write strobe for the baud-rate divisor registers is tied to the **UARTLCRH** register.

UART Line Control (UARTLCRH)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x02C
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								SPS	WLEN		FEN	STP2	EPS	PEN	BRK
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	SPS	R/W	0	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	0	UART Word Length The bits indicate the number of data bits transmitted or received in a frame as follows: Value Description 0x3 8 bits 0x2 7 bits 0x1 6 bits 0x0 5 bits (default)
4	FEN	R/W	0	UART Enable FIFOs If this bit is set to 1, transmit and receive FIFO buffers are enabled (FIFO mode). When cleared to 0, FIFOs are disabled (Character mode). The FIFOs become 1-byte-deep holding registers.

Bit/Field	Name	Type	Reset	Description
3	STP2	R/W	0	UART Two Stop Bits Select If this bit is set to 1, two stop bits are transmitted at the end of a frame. The receive logic does not check for two stop bits being received.
2	EPS	R/W	0	UART Even Parity Select If this bit is set to 1, even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits. When cleared to 0, then odd parity is performed, which checks for an odd number of 1s. This bit has no effect when parity is disabled by the PEN bit.
1	PEN	R/W	0	UART Parity Enable If this bit is set to 1, parity checking and generation is enabled; otherwise, parity is disabled and no parity bit is added to the data frame.
0	BRK	R/W	0	UART Send Break If this bit is set to 1, a Low level is continually output on the UnTX output, after completing transmission of the current character. For the proper execution of the break command, the software must set this bit for at least two frames (character periods). For normal use, this bit must be cleared to 0.

Register 7: UART Control (UARTCTL), offset 0x030

The **UARTCTL** register is the control register. All the bits are cleared on reset except for the `Transmit Enable (TXE)` and `Receive Enable (RXE)` bits, which are set to 1.

To enable the UART module, the `UARTEN` bit must be set to 1. If software requires a configuration change in the module, the `UARTEN` bit must be cleared before the configuration changes are written. If the UART is disabled during a transmit or receive operation, the current transaction is completed prior to the UART stopping.

UART Control (UARTCTL)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x030
Type R/W, reset 0x0000.0300

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	reserved						UARTEN
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO	RO	RO	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	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	RXE	R/W	1	UART Receive Enable If this bit is set to 1, the receive section of the UART is enabled. When the UART is disabled in the middle of a receive, it completes the current character before stopping. **Note:** To enable reception, the `UARTEN` bit must also be set.
8	TXE	R/W	1	UART Transmit Enable If this bit is set to 1, the transmit section of the UART is enabled. When the UART is disabled in the middle of a transmission, it completes the current character before stopping. **Note:** To enable transmission, the `UARTEN` bit must also be set.
7	LBE	R/W	0	UART Loop Back Enable If this bit is set to 1, the `UnTX` path is fed through the `UnRX` path.
6:1	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	UARTEN	R/W	0	UART Enable If this bit is set to 1, the UART is enabled. When the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.

Register 8: UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034

The **UARTIFLS** register is the interrupt FIFO level select register. You can use this register to define the FIFO level at which the `TXRIS` and `RXRIS` bits in the **UARTRIS** register are triggered.

The interrupts are generated based on a transition through a level rather than being based on the level. That is, the interrupts are generated when the fill level progresses through the trigger level. For example, if the receive trigger level is set to the half-way mark, the interrupt is triggered as the module is receiving the 9th character.

Out of reset, the `TXIFLSEL` and `RXIFLSEL` bits are configured so that the FIFOs trigger an interrupt at the half-way mark.

UART Interrupt FIFO Level Select (UARTIFLS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x034
Type R/W, reset 0x0000.0012

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x00	Software should not rely on the value of 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 ≥ 1/8 full 0x1 RX FIFO ≥ ¼ full 0x2 RX FIFO ≥ ½ full (default) 0x3 RX FIFO ≥ ¾ full 0x4 RX FIFO ≥ 7/8 full 0x5-0x7 Reserved

Bit/Field	Name	Type	Reset	Description
2:0	TXIFLSEL	R/W	0x2	UART Transmit Interrupt FIFO Level Select The trigger points for the transmit interrupt are as follows: Value Description 0x0 TX FIFO ≤ 1/8 full 0x1 TX FIFO ≤ ¼ full 0x2 TX FIFO ≤ ½ full (default) 0x3 TX FIFO ≤ ¾ full 0x4 TX FIFO ≤ 7/8 full 0x5-0x7 Reserved

Register 9: UART Interrupt Mask (UARTIM), offset 0x038

The **UARTIM** register is the interrupt mask set/clear register.

On a read, this register gives the current value of the mask on the relevant interrupt. Writing a 1 to a bit allows the corresponding raw interrupt signal to be routed to the interrupt controller. Writing a 0 prevents the raw interrupt signal from being sent to the interrupt controller.

UART Interrupt Mask (UARTIM)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x038
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved					OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM	reserved			
Type	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEIM	R/W	0	UART Overrun Error Interrupt Mask On a read, the current mask for the `OEIM` interrupt is returned. Setting this bit to 1 promotes the `OEIM` interrupt to the interrupt controller.
9	BEIM	R/W	0	UART Break Error Interrupt Mask On a read, the current mask for the `BEIM` interrupt is returned. Setting this bit to 1 promotes the `BEIM` interrupt to the interrupt controller.
8	PEIM	R/W	0	UART Parity Error Interrupt Mask On a read, the current mask for the `PEIM` interrupt is returned. Setting this bit to 1 promotes the `PEIM` interrupt to the interrupt controller.
7	FEIM	R/W	0	UART Framing Error Interrupt Mask On a read, the current mask for the `FEIM` interrupt is returned. Setting this bit to 1 promotes the `FEIM` interrupt to the interrupt controller.
6	RTIM	R/W	0	UART Receive Time-Out Interrupt Mask On a read, the current mask for the `RTIM` interrupt is returned. Setting this bit to 1 promotes the `RTIM` interrupt to the interrupt controller.
5	TXIM	R/W	0	UART Transmit Interrupt Mask On a read, the current mask for the `TXIM` interrupt is returned. Setting this bit to 1 promotes the `TXIM` interrupt to the interrupt controller.

Bit/Field	Name	Type	Reset	Description
4	RXIM	R/W	0	UART Receive Interrupt Mask On a read, the current mask for the `RXIM` interrupt is returned. Setting this bit to 1 promotes the `RXIM` interrupt to the interrupt controller.
3:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 10: UART Raw Interrupt Status (UARTRIS), offset 0x03C

The **UARTRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt. A write has no effect.

UART Raw Interrupt Status (UARTRIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x03C
Type RO, reset 0x0000.000F

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved					OERIS	BERIS	PERIS	FERIS	RTRIS	TXRIS	RXRIS	reserved			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OERIS	RO	0	UART Overrun Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
9	BERIS	RO	0	UART Break Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
8	PERIS	RO	0	UART Parity Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
7	FERIS	RO	0	UART Framing Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
6	RTRIS	RO	0	UART Receive Time-Out Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
5	TXRIS	RO	0	UART Transmit Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
4	RXRIS	RO	0	UART Receive Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
3:0	reserved	RO	0xF	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 11: UART Masked Interrupt Status (UARTMIS), offset 0x040

The **UARTMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

UART Masked Interrupt Status (UARTMIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x040
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved					OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS	reserved			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEMIS	RO	0	UART Overrun Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
9	BEMIS	RO	0	UART Break Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
8	PEMIS	RO	0	UART Parity Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
7	FEMIS	RO	0	UART Framing Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
6	RTMIS	RO	0	UART Receive Time-Out Masked Interrupt Status Gives the masked interrupt state of this interrupt.
5	TXMIS	RO	0	UART Transmit Masked Interrupt Status Gives the masked interrupt state of this interrupt.
4	RXMIS	RO	0	UART Receive Masked Interrupt Status Gives the masked interrupt state of this interrupt.
3:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 12: UART Interrupt Clear (UARTICR), offset 0x044

The **UARTICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt (both raw interrupt and masked interrupt, if enabled) is cleared. A write of 0 has no effect.

UART Interrupt Clear (UARTICR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x044
Type W1C, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved					OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC	reserved			
Type	RO	RO	RO	RO	RO	W1C	W1C	W1C	W1C	W1C	W1C	W1C	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEIC	W1C	0	Overrun Error Interrupt Clear The `OEIC` values are defined as follows: Value Description 0 No effect on the interrupt. 1 Clears interrupt.
9	BEIC	W1C	0	Break Error Interrupt Clear The `BEIC` values are defined as follows: Value Description 0 No effect on the interrupt. 1 Clears interrupt.
8	PEIC	W1C	0	Parity Error Interrupt Clear The `PEIC` values are defined as follows: Value Description 0 No effect on the interrupt. 1 Clears interrupt.

Bit/Field	Name	Type	Reset	Description
7	FEIC	W1C	0	Framing Error Interrupt Clear The `FEIC` values are defined as follows: Value Description 0 No effect on the interrupt. 1 Clears interrupt.
6	RTIC	W1C	0	Receive Time-Out Interrupt Clear The `RTIC` values are defined as follows: Value Description 0 No effect on the interrupt. 1 Clears interrupt.
5	TXIC	W1C	0	Transmit Interrupt Clear The `TXIC` values are defined as follows: Value Description 0 No effect on the interrupt. 1 Clears interrupt.
4	RXIC	W1C	0	Receive Interrupt Clear The `RXIC` values are defined as follows: Value Description 0 No effect on the interrupt. 1 Clears interrupt.
3:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 13: UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 4 (UARTPeriphID4)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0xFD0
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x0000	UART Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 14: UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 5 (UARTPeriphID5)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0xFD4
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x0000	UART Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

Register 15: UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 6 (UARTPeriphID6)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0xFD8
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID6							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x0000	UART Peripheral ID Register[23:16] Can be used by software to identify the presence of this peripheral.

Register 16: UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 7 (UARTPeriphID7)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0xFDC
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID7							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x0000	UART Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

Register 17: UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 0 (UARTPeriphID0)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0xFE0
Type RO, reset 0x0000.0011

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x11	UART Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 18: UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 1 (UARTPeriphID1)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0xFE4
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x00	Software should not rely on the value of a reserved 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 19: UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 2 (UARTPeriphID2)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0xFE8
Type RO, reset 0x0000.0018

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x00	Software should not rely on the value of a reserved 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 20: UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 3 (UARTPeriphID3)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0xFEC
Type RO, reset 0x0000.0001

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x00	Software should not rely on the value of a reserved 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 21: UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 0 (UARTPCellID0)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0xFF0
Type RO, reset 0x0000.000D

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x00	Software should not rely on the value of a reserved 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 22: UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 1 (UARTPCellID1)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0xFF4
Type RO, reset 0x0000.00F0

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

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

Register 23: UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 2 (UARTPCellID2)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0xFF8
Type RO, reset 0x0000.0005

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	UART PrimeCell ID Register[23:16] Provides software a standard cross-peripheral identification system.

Register 24: UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART PrimeCell Identification 3 (UARTPCellID3)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0xFFC
Type RO, reset 0x0000.00B1

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	UART PrimeCell ID Register[31:24] Provides software a standard cross-peripheral identification system.

12 Synchronous Serial Interface (SSI)

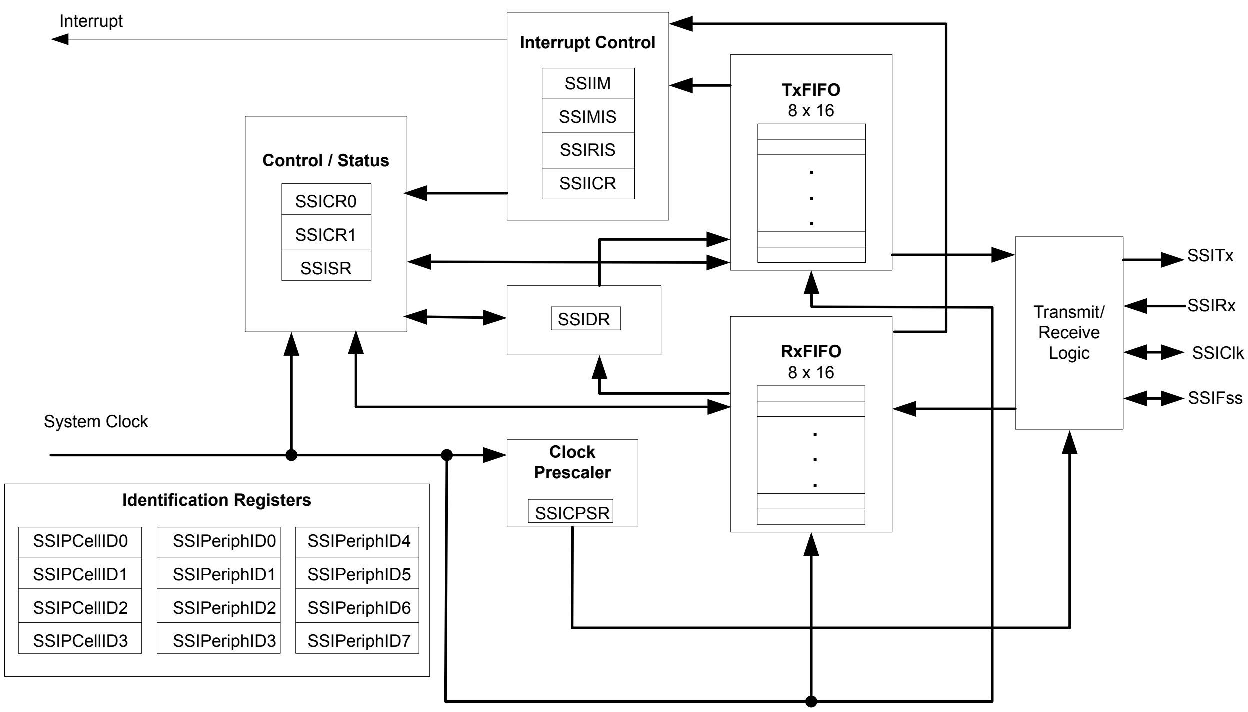
The Stellaris® Synchronous Serial Interface (SSI) is a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces.

The Stellaris® SSI module has the following features:

- Master or slave operation
- Programmable clock bit rate and prescale
- Separate transmit and receive FIFOs, 16 bits wide, 8 locations deep
- Programmable interface operation for Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces
- Programmable data frame size from 4 to 16 bits
- Internal loopback test mode for diagnostic/debug testing

12.1 Block Diagram

Figure 12-1. SSI Module Block Diagram



12.2 Functional Description

The SSI performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information. The transmit and receive paths are buffered with internal FIFO memories allowing up to eight 16-bit values to be stored independently in both transmit and receive modes.

12.2.1 Bit Rate Generation

The SSI includes a programmable bit rate clock divider and prescaler to generate the serial output clock. Bit rates are supported to 2 MHz and higher, although maximum bit rate is determined by peripheral devices.

The serial bit rate is derived by dividing down the 50-MHz input clock. The clock is first divided by an even prescale value CPSDVSR from 2 to 254, which is programmed in the **SSI Clock Prescale (SSICPSR)** register (see page 273). The clock is further divided by a value from 1 to 256, which is 1 + *SCR*, where *SCR* is the value programmed in the **SSI Control0 (SSICR0)** register (see page 266).

The frequency of the output clock SSIClk is defined by:

```
FSSIClk = FSysClk / (CPSDVSR * (1 + SCR))
```

Note that although the SSIClk transmit clock can theoretically be 25 MHz, the module may not be able to operate at that speed. For master mode, the system clock must be at least two times faster than the SSIClk. For slave mode, the system clock must be at least 12 times faster than the SSIClk.

See "Synchronous Serial Interface (SSI)" on page 352 to view SSI timing parameters.

12.2.2 FIFO Operation

12.2.2.1 Transmit FIFO

The common transmit FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. The CPU writes data to the FIFO by writing the **SSI Data (SSIDR)** register (see page 270), 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.

12.2.2.2 Receive FIFO

The common receive FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. Received data from the serial interface is stored in the buffer until read out by the CPU, which accesses the read FIFO by reading the **SSIDR** register.

When configured as a master or slave, serial data received through the SSIRx pin is registered prior to parallel loading into the attached slave or master receive FIFO, respectively.

12.2.3 Interrupts

The SSI can generate interrupts when the following conditions are observed:

- Transmit FIFO service
- Receive FIFO service
- Receive FIFO time-out
- Receive FIFO overrun

All of the interrupt events are ORed together before being sent to the interrupt controller, so the SSI can only generate a single interrupt request to the controller at any given time. You can mask each of the four individual maskable interrupts by setting the appropriate bits in the **SSI Interrupt Mask (SSIIM)** register (see page 274). Setting the appropriate mask bit to 1 enables the interrupt.

Provision of the individual outputs, as well as a combined interrupt output, allows use of either a global interrupt service routine, or modular device drivers to handle interrupts. The transmit and receive dynamic dataflow interrupts have been separated from the status interrupts so that data can be read or written in response to the FIFO trigger levels. The status of the individual interrupt sources can be read from the **SSI Raw Interrupt Status (SSIRIS)** and **SSI Masked Interrupt Status (SSIMIS)** registers (see page 276 and page 277, respectively).

12.2.4 Frame Formats

Each data frame is between 4 and 16 bits long, depending on the size of data programmed, and is transmitted starting with the MSB. There are three basic frame types that can be selected:

- Texas Instruments synchronous serial
- Freescale SPI
- MICROWIRE

For all three formats, the serial clock (`SSIClk`) is held inactive while the SSI is idle, and `SSIClk` transitions at the programmed frequency only during active transmission or reception of data. The idle state of `SSIClk` is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Freescale SPI and MICROWIRE frame formats, the serial frame (`SSIFss`) pin is active Low, and is asserted (pulled down) during the entire transmission of the frame.

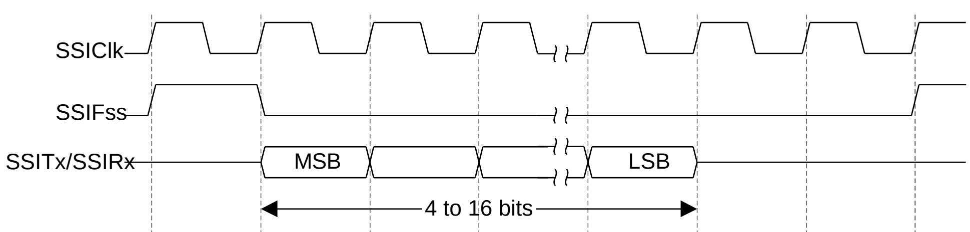
For Texas Instruments synchronous serial frame format, the `SSIFss` pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSI and the off-chip slave device drive their output data on the rising edge of `SSIClk`, and latch data from the other device on the falling edge.

Unlike the full-duplex transmission of the other two frame formats, the MICROWIRE format uses a special master-slave messaging technique, which operates at half-duplex. In this mode, when a frame begins, an 8-bit control message is transmitted to the off-chip slave. During this transmit, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the requested data. The returned data can be 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

12.2.4.1 Texas Instruments Synchronous Serial Frame Format

Figure 12-2 on page 256 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

Figure 12-2. TI Synchronous Serial Frame Format (Single Transfer)

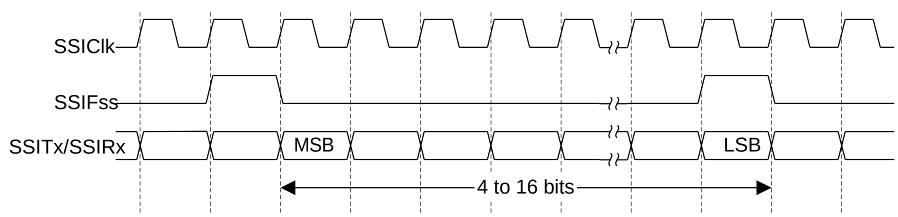


In this mode, SSIClk and SSIFss are forced Low, and the transmit data line SSITx is tristated whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, SSIFss is pulsed High for one SSIClk period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of SSIClk, the MSB of the 4 to 16-bit data frame is shifted out on the SSITx pin. Likewise, the MSB of the received data is shifted onto the SSIRx pin by the off-chip serial slave device.

Both the SSI and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each SSIClk. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SSIClk after the LSB has been latched.

Figure 12-3 on page 257 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

Figure 12-3. TI Synchronous Serial Frame Format (Continuous Transfer)



12.2.4.2 Freescale SPI Frame Format

The Freescale SPI interface is a four-wire interface where the SSIFss signal behaves as a slave select. The main feature of the Freescale SPI format is that the inactive state and phase of the SSIClk signal are programmable through the SPO and SPH bits within the **SSISCR0** control register.

SPO Clock Polarity Bit

When the SPO clock polarity control bit is Low, it produces a steady state Low value on the SSIClk pin. If the SPO bit is High, a steady state High value is placed on the SSIClk pin when data is not being transferred.

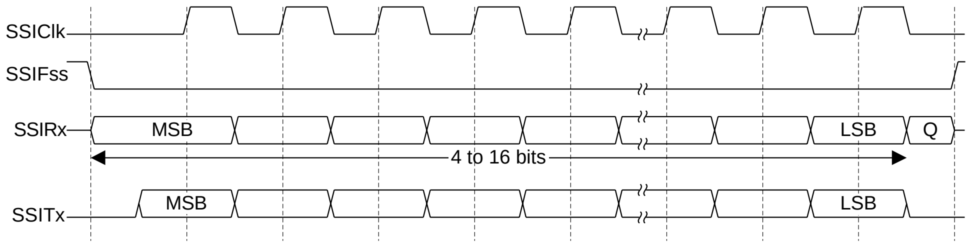
SPH Phase Control Bit

The SPH phase control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the SPH phase control bit is Low, data is captured on the first clock edge transition. If the SPH bit is High, data is captured on the second clock edge transition.

12.2.4.3 Freescale SPI Frame Format with SPO=0 and SPH=0

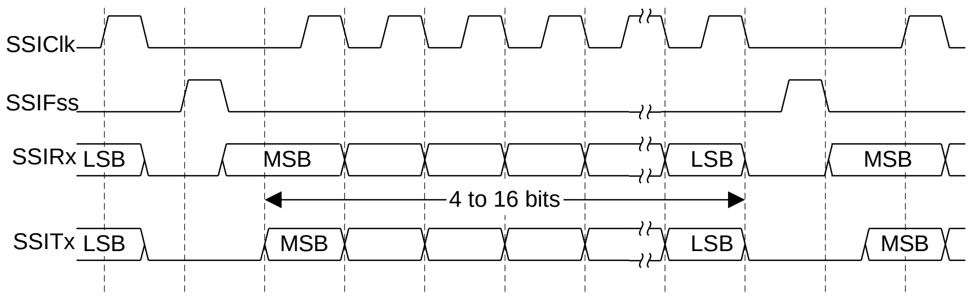
Single and continuous transmission signal sequences for Freescale SPI format with SPO=0 and SPH=0 are shown in Figure 12-4 on page 258 and Figure 12-5 on page 258.

Figure 12-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0



Note: Q is undefined.

Figure 12-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0



In this configuration, during idle periods:

- `SSIClk` is forced Low
- `SSIFss` is forced High
- The transmit data line `SSITx` is arbitrarily forced Low
- When the SSI is configured as a master, it enables the `SSIClk` pad
- When the SSI is configured as a slave, it disables the `SSIClk` pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the `SSIFss` master signal being driven Low. This causes slave data to be enabled onto the `SSIRx` input line of the master. The master `SSITx` output pad is enabled.

One half `SSIClk` period later, valid master data is transferred to the `SSITx` pin. Now that both the master and slave data have been set, the `SSIClk` master clock pin goes High after one further half `SSIClk` period.

The data is now captured on the rising and propagated on the falling edges of the `SSIClk` signal.

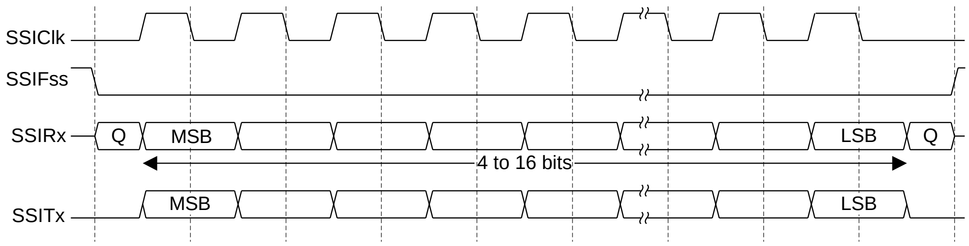
In the case of a single word transmission, after all bits of the data word have been transferred, the `SSIFss` line is returned to its idle High state one `SSIClk` period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the `SSIFss` signal must be pulsed High between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the `SPH` bit is logic zero. Therefore, the master device must raise the `SSIFss` pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the `SSIFss` pin is returned to its idle state one `SSIClk` period after the last bit has been captured.

12.2.4.4 Freescale SPI Frame Format with SPO=0 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=0 and SPH=1 is shown in Figure 12-6 on page 259, which covers both single and continuous transfers.

Figure 12-6. Freescale SPI Frame Format with SPO=0 and SPH=1



Note: Q is undefined.

In this configuration, during idle periods:

- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. The master SSITx output is enabled. After a further one half SSIClk period, both master and slave valid data is enabled onto their respective transmission lines. At the same time, the SSIClk is enabled with a rising edge transition.

Data is then captured on the falling edges and propagated on the rising edges of the SSIClk signal.

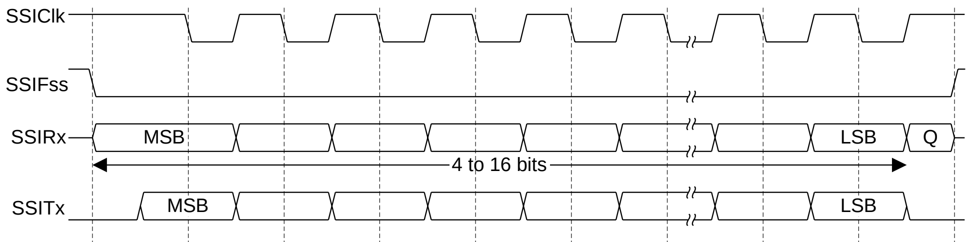
In the case of a single word transfer, after all bits have been transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

For continuous back-to-back transfers, the SSIFss pin is held Low between successive data words and termination is the same as that of the single word transfer.

12.2.4.5 Freescale SPI Frame Format with SPO=1 and SPH=0

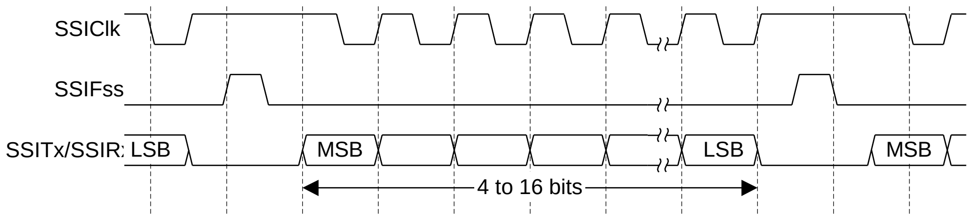
Single and continuous transmission signal sequences for Freescale SPI format with SPO=1 and SPH=0 are shown in Figure 12-7 on page 260 and Figure 12-8 on page 260.

Figure 12-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0



Note: Q is undefined.

Figure 12-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0



In this configuration, during idle periods:

- `SSIClk` is forced High
- `SSIFss` is forced High
- The transmit data line `SSITx` is arbitrarily forced Low
- When the SSI is configured as a master, it enables the `SSIClk` pad
- When the SSI is configured as a slave, it disables the `SSIClk` pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the `SSIFss` master signal being driven Low, which causes slave data to be immediately transferred onto the `SSIRx` line of the master. The master `SSITx` output pad is enabled.

One half period later, valid master data is transferred to the `SSITx` line. Now that both the master and slave data have been set, the `SSIClk` master clock pin becomes Low after one further half `SSIClk` period. This means that data is captured on the falling edges and propagated on the rising edges of the `SSIClk` signal.

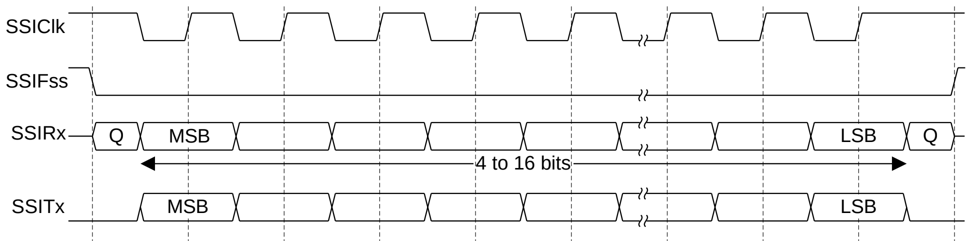
In the case of a single word transmission, after all bits of the data word are transferred, the `SSIFss` line is returned to its idle High state one `SSIClk` period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the `SSIFss` signal must be pulsed High between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the `SPH` bit is logic zero. Therefore, the master device must raise the `SSIFss` pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the `SSIFss` pin is returned to its idle state one `SSIClk` period after the last bit has been captured.

12.2.4.6 Freescale SPI Frame Format with SPO=1 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=1 and SPH=1 is shown in Figure 12-9 on page 261, which covers both single and continuous transfers.

Figure 12-9. Freescale SPI Frame Format with SPO=1 and SPH=1



Note: Q is undefined.

In this configuration, during idle periods:

- `SSIClk` is forced High
- `SSIFss` is forced High
- The transmit data line `SSITx` is arbitrarily forced Low
- When the SSI is configured as a master, it enables the `SSIClk` pad
- When the SSI is configured as a slave, it disables the `SSIClk` pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the `SSIFss` master signal being driven Low. The master `SSITx` output pad is enabled. After a further one-half `SSIClk` period, both master and slave data are enabled onto their respective transmission lines. At the same time, `SSIClk` is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the `SSIClk` signal.

After all bits have been transferred, in the case of a single word transmission, the `SSIFss` line is returned to its idle high state one `SSIClk` period after the last bit has been captured.

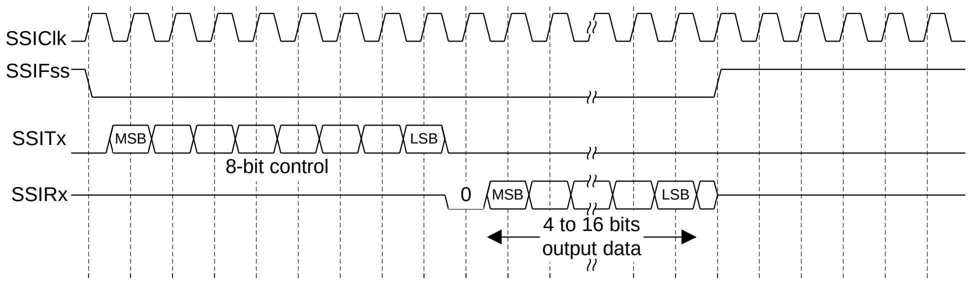
For continuous back-to-back transmissions, the `SSIFss` pin remains in its active Low state, until the final bit of the last word has been captured, and then returns to its idle state as described above.

For continuous back-to-back transfers, the `SSIFss` pin is held Low between successive data words and termination is the same as that of the single word transfer.

12.2.4.7 MICROWIRE Frame Format

Figure 12-10 on page 262 shows the MICROWIRE frame format, again for a single frame. Figure 12-11 on page 263 shows the same format when back-to-back frames are transmitted.

Figure 12-10. MICROWIRE Frame Format (Single Frame)



MICROWIRE format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex, using a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSI to the off-chip slave device. During this transmission, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

In this configuration, during idle periods:

- `SSIClk` is forced Low
- `SSIFss` is forced High
- The transmit data line `SSITx` is arbitrarily forced Low

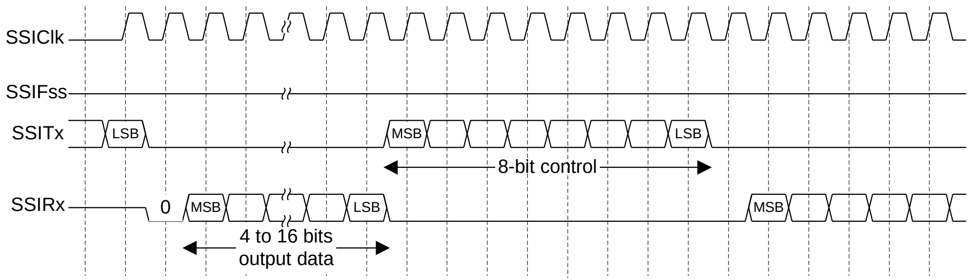
A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of `SSIFss` causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic, and the MSB of the 8-bit control frame to be shifted out onto the `SSITx` pin. `SSIFss` remains Low for the duration of the frame transmission. The `SSIRx` pin remains tristated during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on the rising edge of each `SSIClk`. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state, and the slave responds by transmitting data back to the SSI. Each bit is driven onto the `SSIRx` line on the falling edge of `SSIClk`. The SSI in turn latches each bit on the rising edge of `SSIClk`. At the end of the frame, for single transfers, the `SSIFss` signal is pulled High one clock period after the last bit has been latched in the receive serial shifter, which causes the data to be transferred to the receive FIFO.

Note: The off-chip slave device can tristate the receive line either on the falling edge of `SSIClk` after the LSB has been latched by the receive shifter, or when the `SSIFss` pin goes High.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the `SSIFss` line is continuously asserted (held Low) and transmission of data occurs back-to-back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge of `SSIClk`, after the LSB of the frame has been latched into the SSI.

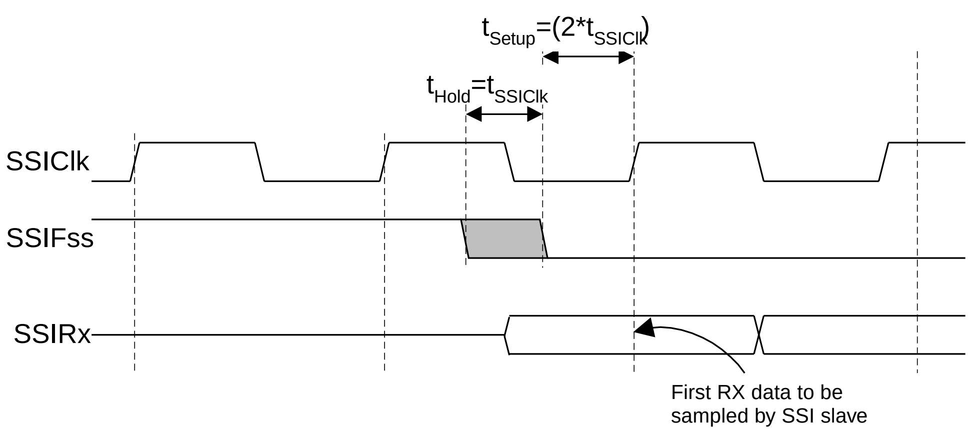
Figure 12-11. MICROWIRE Frame Format (Continuous Transfer)



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 12-12 on page 263 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 12-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements



12.3 Initialization and Configuration

To use the SSI, its peripheral clock must be enabled by setting the `SSI` bit in the **RCGC1** register.

For each of the frame formats, the SSI is configured using the following steps:

1. Ensure that the `SSE` bit in the **SSICR1** register is disabled before making any configuration changes.
2. Select whether the SSI is a master or slave:
   a. For master operations, set the **SSICR1** register to 0x0000.0000.
   b. For slave mode (output enabled), set the **SSICR1** register to 0x0000.0004.
   c. For slave mode (output disabled), set the **SSICR1** register to 0x0000.000C.
3. Configure the clock prescale divisor by writing the **SSICPSR** register.

4. Write the **SSICR0** register with the following configuration:
   - Serial clock rate (SCR)
   - Desired clock phase/polarity, if using Freescale SPI mode (SPH and SPO)
   - The protocol mode: Freescale SPI, TI SSF, MICROWIRE (FRF)
   - The data size (DSS)
5. Enable the SSI by setting the SSE bit in the **SSICR1** register.

As an example, assume the SSI must be configured to operate with the following parameters:

- Master operation
- Freescale SPI mode (SPO=1, SPH=1)
- 1 Mbps bit rate
- 8 data bits

Assuming the system clock is 20 MHz, the bit rate calculation would be:

```
FSSIClk = FSysClk / (CPSDVSR * (1 + SCR))
1x106 = 20x106 / (CPSDVSR * (1 + SCR))
```

In this case, if CPSDVSR=2, SCR must be 9.

The configuration sequence would be as follows:

1. Ensure that the SSE bit in the **SSICR1** register is disabled.
2. Write the **SSICR1** register with a value of 0x0000.0000.
3. Write the **SSICPSR** register with a value of 0x0000.0002.
4. Write the **SSICR0** register with a value of 0x0000.09C7.
5. The SSI is then enabled by setting the SSE bit in the **SSICR1** register to 1.

12.4 Register Map

Table 12-1 on page 264 lists the SSI registers. The offset listed is a hexadecimal increment to the register's address, relative to that SSI module's base address:

- SSI0: 0x4000.8000

Note: The SSI must be disabled (see the SSE bit in the **SSICR1** register) before any of the control registers are reprogrammed.

Table 12-1. SSI Register Map

Offset	Name	Type	Reset	Description	See page
0x000	SSICR0	R/W	0x0000.0000	SSI Control 0	266

Offset	Name	Type	Reset	Description	See page
0x004	SSICR1	R/W	0x0000.0000	SSI Control 1	268
0x008	SSIDR	R/W	0x0000.0000	SSI Data	270
0x00C	SSISR	RO	0x0000.0003	SSI Status	271
0x010	SSICPSR	R/W	0x0000.0000	SSI Clock Prescale	273
0x014	SSIIM	R/W	0x0000.0000	SSI Interrupt Mask	274
0x018	SSIRIS	RO	0x0000.0008	SSI Raw Interrupt Status	276
0x01C	SSIMIS	RO	0x0000.0000	SSI Masked Interrupt Status	277
0x020	SSIICR	W1C	0x0000.0000	SSI Interrupt Clear	278
0xFD0	SSIPeriphID4	RO	0x0000.0000	SSI Peripheral Identification 4	279
0xFD4	SSIPeriphID5	RO	0x0000.0000	SSI Peripheral Identification 5	280
0xFD8	SSIPeriphID6	RO	0x0000.0000	SSI Peripheral Identification 6	281
0xFDC	SSIPeriphID7	RO	0x0000.0000	SSI Peripheral Identification 7	282
0xFE0	SSIPeriphID0	RO	0x0000.0022	SSI Peripheral Identification 0	283
0xFE4	SSIPeriphID1	RO	0x0000.0000	SSI Peripheral Identification 1	284
0xFE8	SSIPeriphID2	RO	0x0000.0018	SSI Peripheral Identification 2	285
0xFEC	SSIPeriphID3	RO	0x0000.0001	SSI Peripheral Identification 3	286
0xFF0	SSIPCellID0	RO	0x0000.000D	SSI PrimeCell Identification 0	287
0xFF4	SSIPCellID1	RO	0x0000.00F0	SSI PrimeCell Identification 1	288
0xFF8	SSIPCellID2	RO	0x0000.0005	SSI PrimeCell Identification 2	289
0xFFC	SSIPCellID3	RO	0x0000.00B1	SSI PrimeCell Identification 3	290

12.5 Register Descriptions

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

Register 1: SSI Control 0 (SSICR0), offset 0x000

SSICR0 is control register 0 and contains bit fields that control various functions within the SSI module. Functionality such as protocol mode, clock rate, and data size are configured in this register.

SSI Control 0 (SSICR0)

SSI0 base: 0x4000.8000
Offset 0x000
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	SCR								SPH	SPO	FRF		DSS			
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:8	SCR	R/W	0x0000	SSI Serial Clock Rate The value SCR is used to generate the transmit and receive bit rate of the SSI. The bit rate is: `BR=FSSIClk/(CPSDVSR * (1 + SCR))` where CPSDVSR is an even value from 2-254 programmed in the **SSICPSR** register, and SCR is a value from 0-255.
7	SPH	R/W	0	SSI Serial Clock Phase This bit is only applicable to the Freescale SPI Format. The SPH control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the SPH bit is 0, data is captured on the first clock edge transition. If SPH is 1, data is captured on the second clock edge transition.
6	SPO	R/W	0	SSI Serial Clock Polarity This bit is only applicable to the Freescale SPI Format. When the SPO bit is 0, it produces a steady state Low value on the SSIClk pin. If SPO is 1, a steady state High value is placed on the SSIClk pin when data is not being transferred.

Bit/Field	Name	Type	Reset	Description
5:4	FRF	R/W	0x0	SSI Frame Format Select The FRF values are defined as follows: Value Frame Format 0x0 Freescale SPI Frame Format 0x1 Texas Intruments Synchronous Serial Frame Format 0x2 MICROWIRE Frame Format 0x3 Reserved
3:0	DSS	R/W	0x00	SSI Data Size Select The DSS values are defined as follows: Value Data Size 0x0-0x2 Reserved 0x3 4-bit data 0x4 5-bit data 0x5 6-bit data 0x6 7-bit data 0x7 8-bit data 0x8 9-bit data 0x9 10-bit data 0xA 11-bit data 0xB 12-bit data 0xC 13-bit data 0xD 14-bit data 0xE 15-bit data 0xF 16-bit data

Register 2: SSI Control 1 (SSICR1), offset 0x004

SSICR1 is control register 1 and contains bit fields that control various functions within the SSI module. Master and slave mode functionality is controlled by this register.

SSI Control 1 (SSICR1)

SSI0 base: 0x4000.8000
Offset 0x004
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	SOD	R/W	0	SSI Slave Mode Output Disable This bit is relevant only in the Slave mode (`MS=1`). In multiple-slave systems, it is possible for the SSI master to broadcast a message to all slaves in the system while ensuring that only one slave drives data onto the serial output line. In such systems, the TXD lines from multiple slaves could be tied together. To operate in such a system, the `SOD` bit can be configured so that the SSI slave does not drive the `SSITx` pin. The `SOD` values are defined as follows: Value Description 0 SSI can drive `SSITx` output in Slave Output mode. 1 SSI must not drive the `SSITx` output in Slave mode.
2	MS	R/W	0	SSI Master/Slave Select This bit selects Master or Slave mode and can be modified only when SSI is disabled (`SSE=0`). The `MS` values are defined as follows: Value Description 0 Device configured as a master. 1 Device configured as a slave.

Bit/Field	Name	Type	Reset	Description
1	SSE	R/W	0	SSI Synchronous Serial Port Enable Setting this bit enables SSI operation. The SSE values are defined as follows: Value Description 0 SSI operation disabled. 1 SSI operation enabled. **Note:** This bit must be set to 0 before any control registers are reprogrammed.
0	LBM	R/W	0	SSI Loopback Mode Setting this bit enables Loopback Test mode. The LBM values are defined as follows: Value Description 0 Normal serial port operation enabled. 1 Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.

Register 3: SSI Data (SSIDR), offset 0x008

SSIDR is the data register and is 16-bits wide. When **SSIDR** is read, the entry in the receive FIFO (pointed to by the current FIFO read pointer) is accessed. As data values are removed by the SSI receive logic from the incoming data frame, they are placed into the entry in the receive FIFO (pointed to by the current FIFO write pointer).

When **SSIDR** is written to, the entry in the transmit FIFO (pointed to by the write pointer) is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. It is loaded into the transmit serial shifter, then serially shifted out onto the SSITx pin at the programmed bit rate.

When a data size of less than 16 bits is selected, the user must right-justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 bits is automatically right-justified in the receive buffer.

When the SSI is programmed for MICROWIRE frame format, the default size for transmit data is eight bits (the most significant byte is ignored). The receive data size is controlled by the programmer. The transmit FIFO and the receive FIFO are not cleared even when the SSE bit in the **SSICR1** register is set to zero. This allows the software to fill the transmit FIFO before enabling the SSI.

SSI Data (SSIDR)

SSI0 base: 0x4000.8000
Offset 0x008
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DATA	R/W	0x0000	SSI Receive/Transmit Data A read operation reads the receive FIFO. A write operation writes the transmit FIFO. Software must right-justify data when the SSI is programmed for a data size that is less than 16 bits. Unused bits at the top are ignored by the transmit logic. The receive logic automatically right-justifies the data.

Register 4: SSI Status (SSISR), offset 0x00C

SSISR is a status register that contains bits that indicate the FIFO fill status and the SSI busy status.

SSI Status (SSISR)

SSI0 base: 0x4000.8000
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	R0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	BSY	RO	0	SSI Busy Bit The `BSY` values are defined as follows: Value Description 0 SSI is idle. 1 SSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.
3	RFF	RO	0	SSI Receive FIFO Full The `RFF` values are defined as follows: Value Description 0 Receive FIFO is not full. 1 Receive FIFO is full.
2	RNE	RO	0	SSI Receive FIFO Not Empty The `RNE` values are defined as follows: Value Description 0 Receive FIFO is empty. 1 Receive FIFO is not empty.
1	TNF	RO	1	SSI Transmit FIFO Not Full The `TNF` values are defined as follows: Value Description 0 Transmit FIFO is full. 1 Transmit FIFO is not full.

Bit/Field	Name	Type	Reset	Description
0	TFE	R0	1	SSI Transmit FIFO Empty The TFE values are defined as follows: Value Description 0 Transmit FIFO is not empty. 1 Transmit FIFO is empty.

Register 5: SSI Clock Prescale (SSICPSR), offset 0x010

SSICPSR is the clock prescale register and specifies the division factor by which the system clock must be internally divided before further use.

The value programmed into this register must be an even number between 2 and 254. The least-significant bit of the programmed number is hard-coded to zero. If an odd number is written to this register, data read back from this register has the least-significant bit as zero.

SSI Clock Prescale (SSICPSR)

SSI0 base: 0x4000.8000
Offset 0x010
Type R/W, reset 0x0000.0000

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

On a read, this register gives the current value of the mask on the relevant interrupt. A write of 1 to the particular bit sets the mask, enabling the interrupt to be read. A write of 0 clears the corresponding mask.

SSI Interrupt Mask (SSIIM)

SSI0 base: 0x4000.8000
Offset 0x014
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXIM	R/W	0	SSI Transmit FIFO Interrupt Mask The `TXIM` values are defined as follows: Value Description 0 TX FIFO half-full or less condition interrupt is masked. 1 TX FIFO half-full or less condition interrupt is not masked.
2	RXIM	R/W	0	SSI Receive FIFO Interrupt Mask The `TFE` values are defined as follows: Value Description 0 RX FIFO half-full or more condition interrupt is masked. 1 RX FIFO half-full or more condition interrupt is not masked.
1	RTIM	R/W	0	SSI Receive Time-Out Interrupt Mask The `RTIM` values are defined as follows: Value Description 0 RX FIFO time-out interrupt is masked. 1 RX FIFO time-out interrupt is not masked.

Bit/Field	Name	Type	Reset	Description
0	RORIM	R/W	0	SSI Receive Overrun Interrupt Mask The RORIM values are defined as follows: Value Description 0 RX FIFO overrun interrupt is masked. 1 RX FIFO overrun interrupt is not masked.

Register 7: SSI Raw Interrupt Status (SSIRIS), offset 0x018

The **SSIRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

SSI Raw Interrupt Status (SSIRIS)

SSI0 base: 0x4000.8000
Offset 0x018
Type RO, reset 0x0000.0008

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXRIS	RO	1	SSI Transmit FIFO Raw Interrupt Status Indicates that the transmit FIFO is half full or less, when set.
2	RXRIS	RO	0	SSI Receive FIFO Raw Interrupt Status Indicates that the receive FIFO is half full or more, when set.
1	RTRIS	RO	0	SSI Receive Time-Out Raw Interrupt Status Indicates that the receive time-out has occurred, when set.
0	RORRIS	RO	0	SSI Receive Overrun Raw Interrupt Status Indicates that the receive FIFO has overflowed, when set.

Register 8: SSI Masked Interrupt Status (SSIMIS), offset 0x01C

The **SSIMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

SSI Masked Interrupt Status (SSIMIS)

SSI0 base: 0x4000.8000
Offset 0x01C
Type RO, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXMIS	RO	0	SSI Transmit FIFO Masked Interrupt Status Indicates that the transmit FIFO is half full or less, when set.
2	RXMIS	RO	0	SSI Receive FIFO Masked Interrupt Status Indicates that the receive FIFO is half full or more, when set.
1	RTMIS	RO	0	SSI Receive Time-Out Masked Interrupt Status Indicates that the receive time-out has occurred, when set.
0	RORMIS	RO	0	SSI Receive Overrun Masked Interrupt Status Indicates that the receive FIFO has overflowed, when set.

Register 9: SSI Interrupt Clear (SSIICR), offset 0x020

The **SSIICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

SSI Interrupt Clear (SSIICR)

SSI0 base: 0x4000.8000
Offset 0x020
Type W1C, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														RTIC	RORIC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	RTIC	W1C	0	SSI Receive Time-Out Interrupt Clear The `RTIC` values are defined as follows: Value Description 0 No effect on interrupt. 1 Clears interrupt.
0	RORIC	W1C	0	SSI Receive Overrun Interrupt Clear The `RORIC` values are defined as follows: Value Description 0 No effect on interrupt. 1 Clears interrupt.

Register 10: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0

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

SSI Peripheral Identification 4 (SSIPeriphID4)

SSI0 base: 0x4000.8000
Offset 0xFD0
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID4							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Register 11: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4

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

SSI Peripheral Identification 5 (SSIPeriphID5)

SSI0 base: 0x4000.8000
Offset 0xFD4
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	SSI Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

Register 12: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8

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

SSI Peripheral Identification 6 (SSIPeriphID6)

SSI0 base: 0x4000.8000
Offset 0xFD8
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID6							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Register 13: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC

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

SSI Peripheral Identification 7 (SSIPeriphID7)

SSI0 base: 0x4000.8000
Offset 0xFDC
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID7							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Register 14: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0

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

SSI Peripheral Identification 0 (SSIPeriphID0)

SSI0 base: 0x4000.8000
Offset 0xFE0
Type RO, reset 0x0000.0022

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x22	SSI Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 15: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4

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

SSI Peripheral Identification 1 (SSIPeriphID1)

SSI0 base: 0x4000.8000
Offset 0xFE4
Type RO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	SSI Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 16: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8

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

SSI Peripheral Identification 2 (SSIPeriphID2)

SSI0 base: 0x4000.8000
Offset 0xFE8
Type RO, reset 0x0000.0018

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	SSI Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 17: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC

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

SSI Peripheral Identification 3 (SSIPeriphID3)

SSI0 base: 0x4000.8000
Offset 0xFEC
Type RO, reset 0x0000.0001

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								PID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

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

Register 18: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0

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

SSI PrimeCell Identification 0 (SSIPCellID0)

SSI0 base: 0x4000.8000
Offset 0xFF0
Type RO, reset 0x0000.000D

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID0							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

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

Register 19: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4

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

SSI PrimeCell Identification 1 (SSIPCellID1)

SSI0 base: 0x4000.8000
Offset 0xFF4
Type RO, reset 0x0000.00F0

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID1							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0

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

Register 20: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8

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

SSI PrimeCell Identification 2 (SSIPCellID2)

SSI0 base: 0x4000.8000
Offset 0xFF8
Type RO, reset 0x0000.0005

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID2							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

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

Register 21: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC

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

SSI PrimeCell Identification 3 (SSIPCellID3)

SSI0 base: 0x4000.8000
Offset 0xFFC
Type RO, reset 0x0000.00B1

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CID3							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved 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.

13 Inter-Integrated Circuit (I^{2}C) Interface

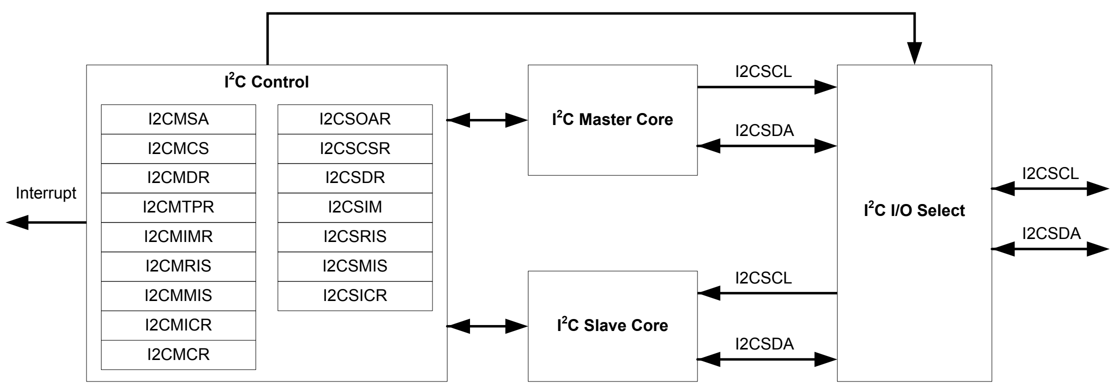
The Inter-Integrated Circuit (I^{2}C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL), and interfaces to external I^{2}C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I^{2}C bus may also be used for system testing and diagnostic purposes in product development and manufacture. The LM3S600 microcontroller includes one I^{2}C module, providing the ability to interact (both send and receive) with other I^{2}C devices on the bus.

Devices on the I^{2}C bus can be designated as either a master or a slave. The Stellaris® I^{2}C module supports both sending and receiving data as either a master or a slave, and also supports the simultaneous operation as both a master and a slave. There are a total of four I^{2}C modes: Master Transmit, Master Receive, Slave Transmit, and Slave Receive. The Stellaris® I^{2}C module can operate at two speeds: Standard (100 Kbps) and Fast (400 Kbps).

Both the I^{2}C master and slave can generate interrupts; the I^{2}C master generates interrupts when a transmit or receive operation completes (or aborts due to an error) and the I^{2}C slave generates interrupts when data has been sent or requested by a master.

13.1 Block Diagram

Figure 13-1. I^{2}C Block Diagram

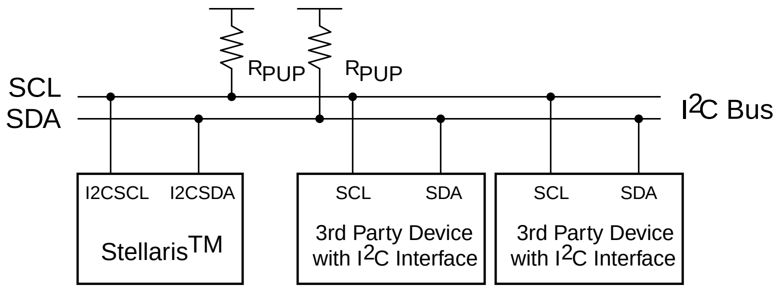


13.2 Functional Description

I^{2}C module is comprised of both master and slave functions which are implemented as separate peripherals. For proper operation, the SDA and SCL pins must be connected to bi-directional open-drain pads. A typical I^{2}C bus configuration is shown in Figure 13-2 on page 292.

See "I^{2}C" on page 351 for I^{2}C timing diagrams.

Figure 13-2. I2C Bus Configuration



13.2.1 I2C Bus Functional Overview

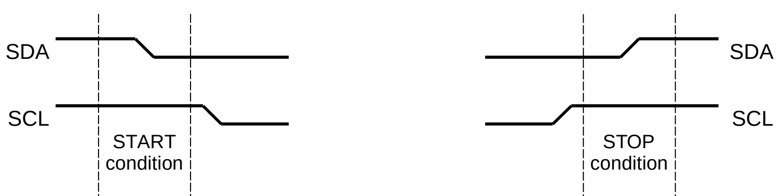
The I2C bus uses only two signals: SDA and SCL, named `I2CSDA` and `I2CSCL` on Stellaris® microcontrollers. SDA is the bi-directional serial data line and SCL is the bi-directional serial clock line. The bus is considered idle when both lines are high.

Every transaction on the I2C bus is nine bits long, consisting of eight data bits and a single acknowledge bit. The number of bytes per transfer (defined as the time between a valid START and STOP condition, described in "START and STOP Conditions" on page 292) 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.

13.2.1.1 START and STOP Conditions

The protocol of the I2C bus defines two states to begin and end a transaction: START and STOP. A high-to-low transition on the SDA line while the SCL is high is defined as a START condition, and a low-to-high transition on the SDA line while SCL is high is defined as a STOP condition. The bus is considered busy after a START condition and free after a STOP condition. See Figure 13-3 on page 292.

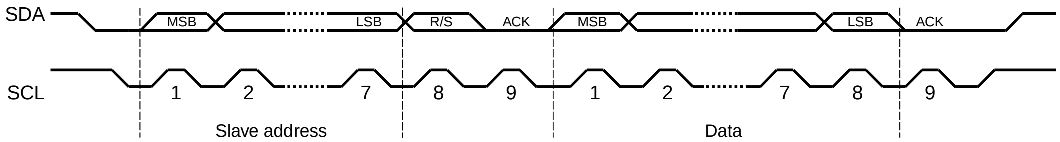
Figure 13-3. START and STOP Conditions



13.2.1.2 Data Format with 7-Bit Address

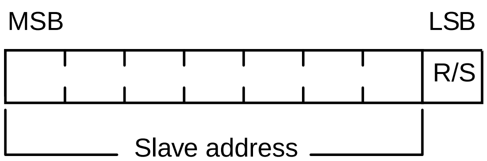
Data transfers follow the format shown in Figure 13-4 on page 293. After the START condition, a slave address is sent. This address is 7-bits long followed by an eighth bit, which is a data direction bit (R/S bit in the **I2CMSA** register). A zero indicates a transmit operation (send), and a one indicates a request for data (receive). A data transfer is always terminated by a STOP condition generated by the master, however, a master can initiate communications with another device on the bus by generating a repeated START condition and addressing another slave without first generating a STOP condition. Various combinations of receive/send formats are then possible within a single transfer.

Figure 13-4. Complete Data Transfer with a 7-Bit Address



The first seven bits of the first byte make up the slave address (see Figure 13-5 on page 293). The eighth bit determines the direction of the message. A zero in the R/S position of the first byte means that the master will write (send) data to the selected slave, and a one in this position means that the master will receive data from the slave.

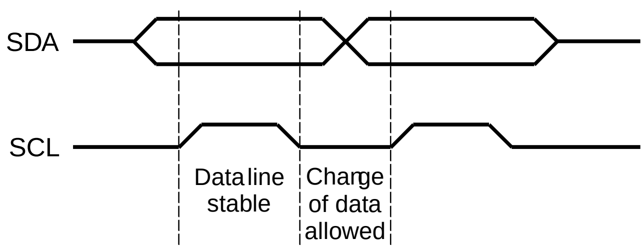
Figure 13-5. R/S Bit in First Byte



13.2.1.3 Data Validity

The data on the SDA line must be stable during the high period of the clock, and the data line can only change when SCL is low (see Figure 13-6 on page 293).

Figure 13-6. Data Validity During Bit Transfer on the I²C Bus



13.2.1.4 Acknowledge

All bus transactions have a required acknowledge clock cycle that is generated by the master. During the acknowledge cycle, the transmitter (which can be the master or slave) releases the SDA line. To acknowledge the transaction, the receiver must pull down SDA during the acknowledge clock cycle. The data sent out by the receiver during the acknowledge cycle must comply with the data validity requirements described in “Data Validity” on page 293.

When a slave receiver does not acknowledge the slave address, SDA must be left high by the slave so that the master can generate a STOP condition and abort the current transfer. If the master device is acting as a receiver during a transfer, it is responsible for acknowledging each transfer made by the slave. Since the master controls the number of bytes in the transfer, it signals the end of data to the slave transmitter by not generating an acknowledge on the last data byte. The slave transmitter must then release SDA to allow the master to generate the STOP or a repeated START condition.

13.2.1.5 Arbitration

A master may start a transfer only if the bus is idle. It's possible for two or more masters to generate a START condition within minimum hold time of the START condition. In these situations, an arbitration scheme takes place on the SDA line, while SCL is high. During arbitration, the first of the competing master devices to place a '1' (high) on SDA while another master transmits a '0' (low) will switch off its data output stage and retire until the bus is idle again.

Arbitration can take place over several bits. Its first stage is a comparison of address bits, and if both masters are trying to address the same device, arbitration continues on to the comparison of data bits.

13.2.2 Available Speed Modes

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 **I^2C Master Timer Period (I2CMTPR)** register (see page 311).

The I^2C clock period is calculated as follows:

```
SCL_PERIOD = 2*(1 + TIMER_PRD)*(SCL_LP + SCL_HP)*CLK_PRD
```

For example:

```
CLK_PRD = 50 ns
TIMER_PRD = 2
SCL_LP=6
SCL_HP=4
```

yields a SCL frequency of:

```
1/T = 333 Khz
```

Table 13-1 on page 294 gives examples of timer period, system clock, and speed mode (Standard or Fast).

Table 13-1. 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
33Mhz	0x10	97.1 Kbps	0x04	330 Kbps
40Mhz	0x13	100 Kbps	0x04	400 Kbps

System Clock	Timer Period	Standard Mode	Timer Period	Fast Mode
50Mhz	0x18	100 Kbps	0x06	357 Kbps

13.2.3 Interrupts

The I²C can generate interrupts when the following conditions are observed:

- Master transaction completed
- Master transaction error
- Slave transaction received
- Slave transaction requested

There is a separate interrupt signal for the I²C master and I²C modules. While both modules can generate interrupts for multiple conditions, only a single interrupt signal is sent to the interrupt controller.

13.2.3.1 I²C Master Interrupts

The I²C master module generates an interrupt when a transaction completes (either transmit or receive), or when an error occurs during a transaction. To enable the I²C master interrupt, software must write a '1' to the **I²C Master Interrupt Mask (I2CMIMR)** register. When an interrupt condition is met, software must check the ERROR bit in the **I²C Master Control/Status (I2CMCS)** register to verify that an error didn't occur during the last transaction. An error condition is asserted if the last transaction wasn't acknowledge by the slave or if the master was forced to give up ownership of the bus due to a lost arbitration round with another master. If an error is not detected, the application can proceed with the transfer. The interrupt is cleared by writing a '1' to the **I²C Master Interrupt Clear (I2CMICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I²C Master Raw Interrupt Status (I2CMRIS)** register.

13.2.3.2 I²C Slave Interrupts

The slave module generates interrupts as it receives requests from an I²C master. To enable the I²C slave interrupt, write a '1' to the **I²C Slave Interrupt Mask (I2CSIMR)** register. Software determines whether the module should write (transmit) or read (receive) data from the **I²C Slave Data (I2CSDR)** register, by checking the RREQ and TREQ bits of the **I²C Slave Control/Status (I2CSCSR)** register. If the slave module is in receive mode and the first byte of a transfer is received, the FBR bit is set along with the RREQ bit. The interrupt is cleared by writing a '1' to the **I²C Slave Interrupt Clear (I2CSICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I²C Slave Raw Interrupt Status (I2CSRIS)** register.

13.2.4 Loopback Operation

The I²C modules can be placed into an internal loopback mode for diagnostic or debug work. This is accomplished by setting the LPBK bit in the **I²C Master Configuration (I2CMCR)** register. In loopback mode, the SDA and SCL signals from the master and slave modules are tied together.

13.2.5 Command Sequence Flow Charts

This section details the steps required to perform the various I2C transfer types in both master and slave mode.

13.2.5.1 I2C Master Command Sequences

The figures that follow show the command sequences available for the I2C master.

Figure 13-7. Master Single SEND

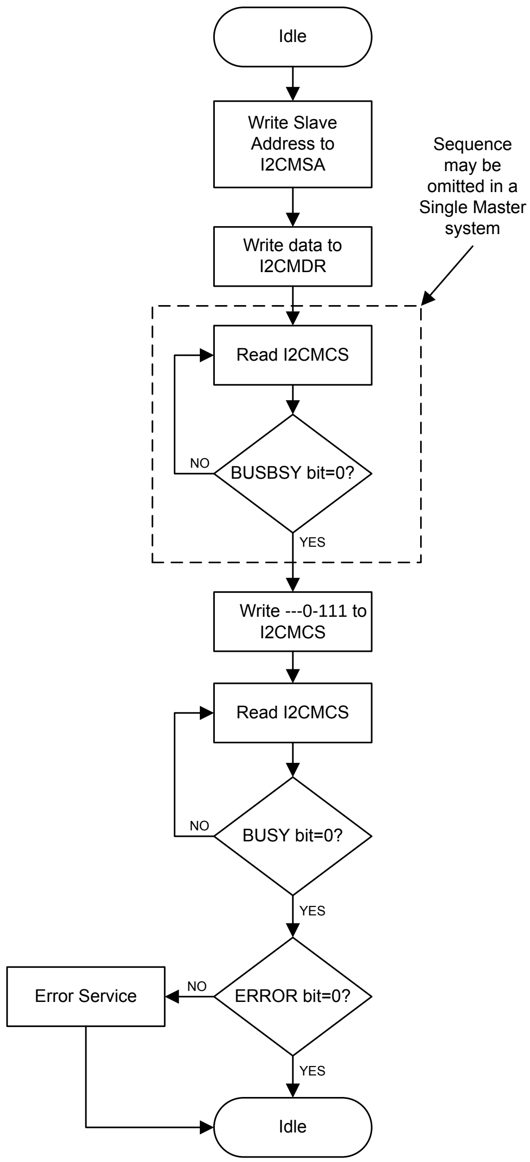


Figure 13-8. Master Single RECEIVE

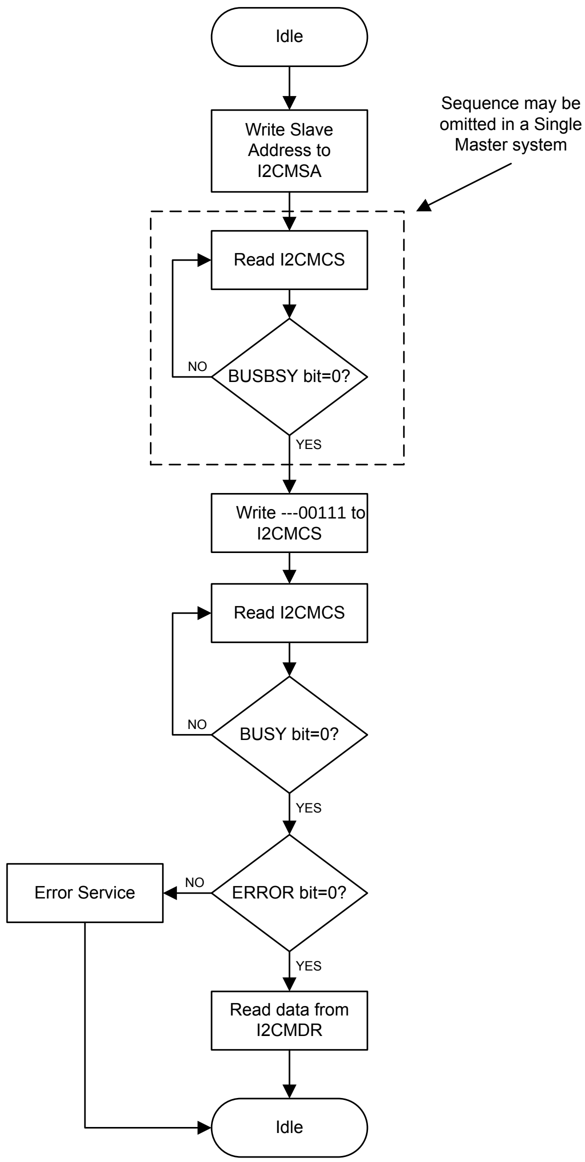


Figure 13-9. Master Burst SEND

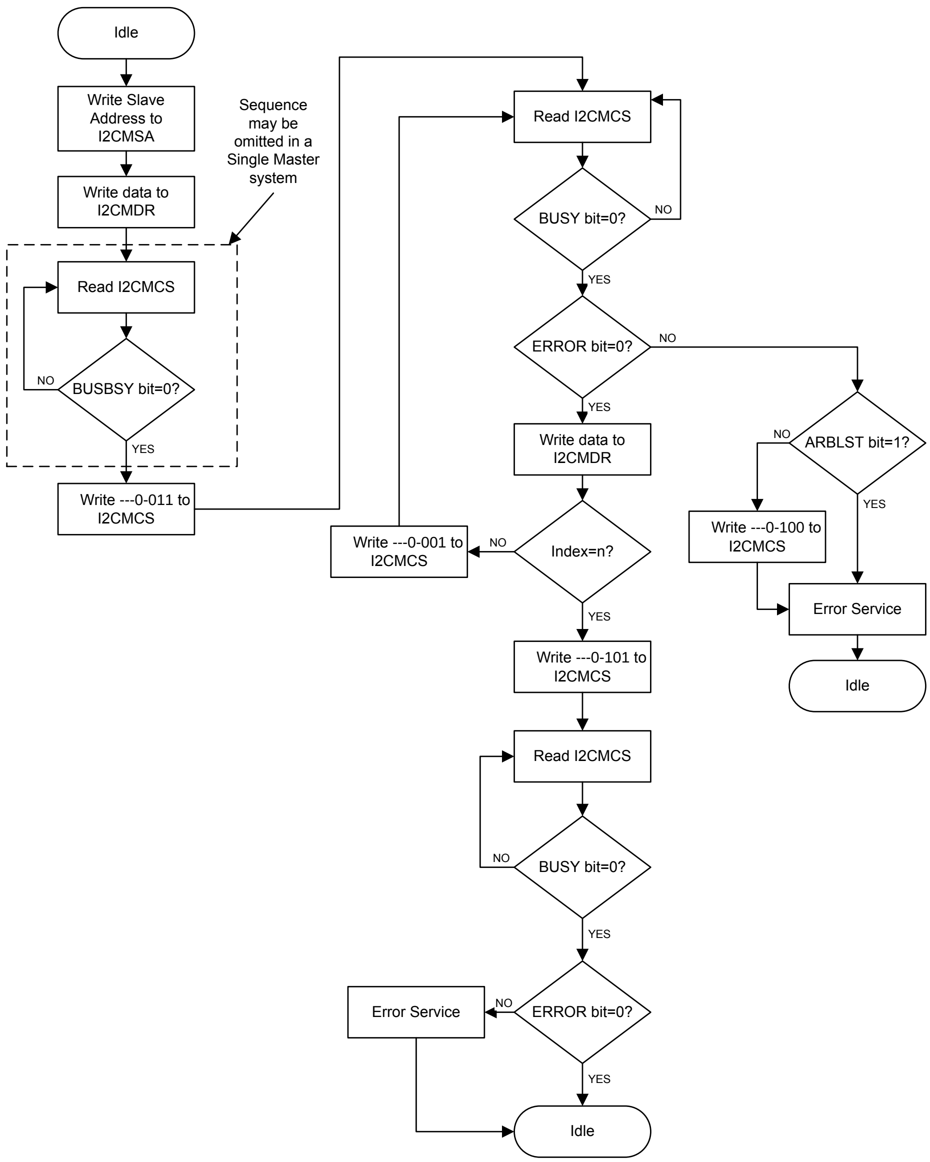


Figure 13-10. Master Burst RECEIVE

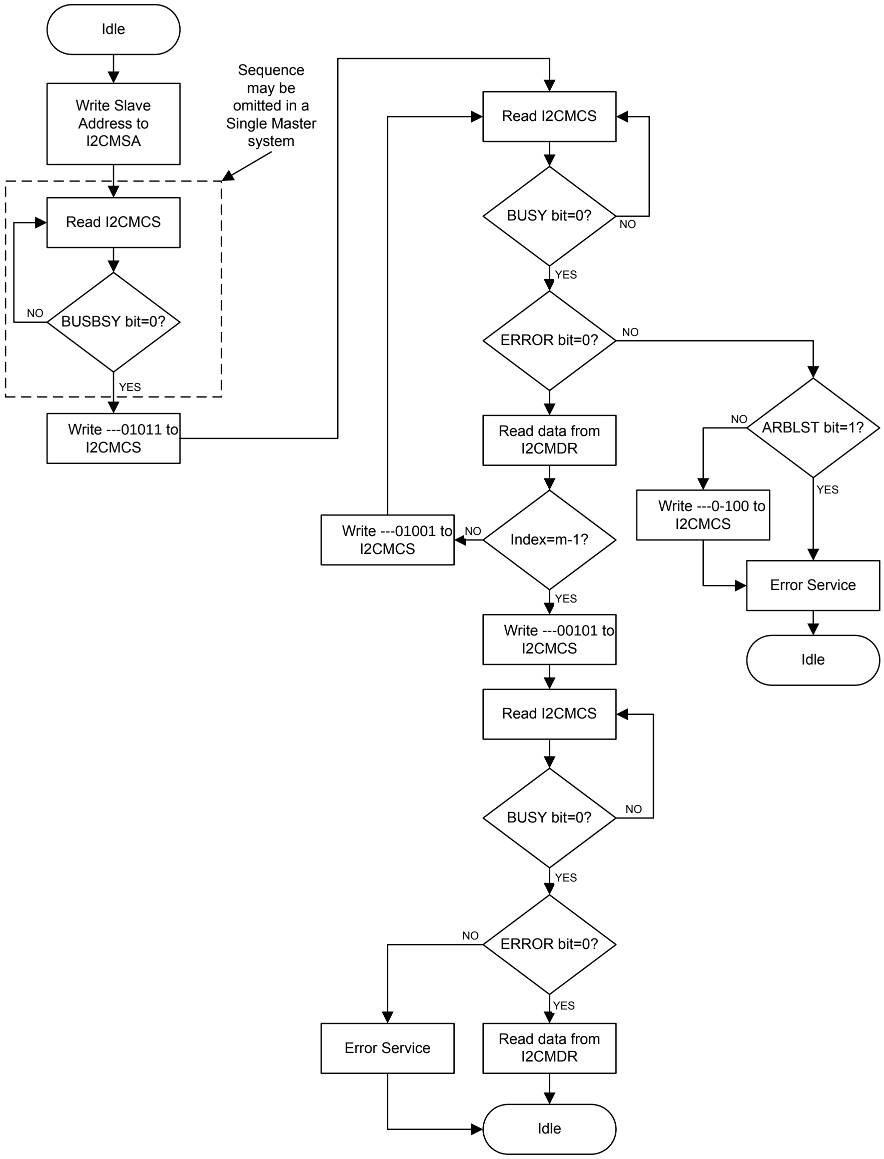


Figure 13-11. Master Burst RECEIVE after Burst SEND

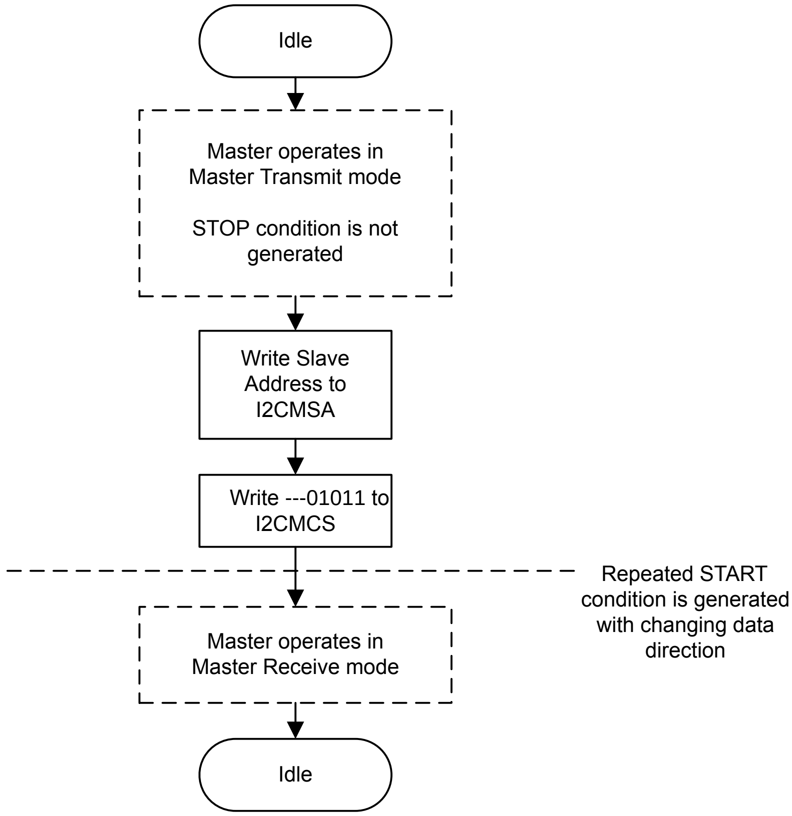
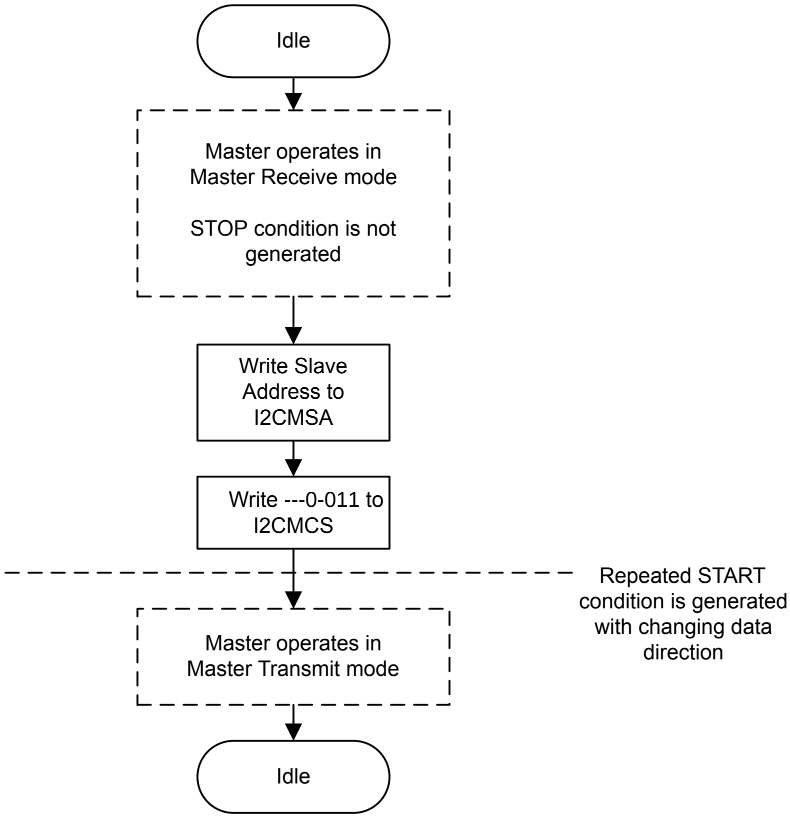


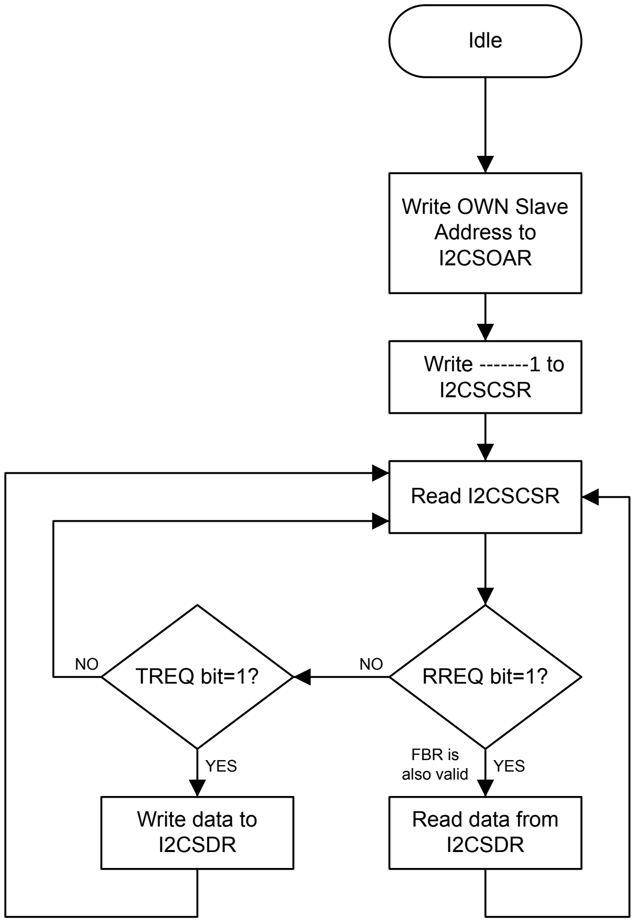
Figure 13-12. Master Burst SEND after Burst RECEIVE



13.2.5.2 I²C Slave Command Sequences

Figure 13-13 on page 302 presents the command sequence available for the I²C slave.

Figure 13-13. Slave Command Sequence



13.3 Initialization and Configuration

The following example shows how to configure the I²C module to send a single byte as a master. This assumes the system clock is 20 MHz.

1. Enable the I²C clock by writing a value of 0x0000.1000 to the **RCGC1** register in the System Control module.

2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module.

3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register. Also, be sure to enable the same pins for Open Drain operation.

4. Initialize the I²C Master by writing the **I2CMCR** register with a value of 0x0000.0020.

5. Set the desired SCL clock speed of 100 Kbps by writing the **I2CMTPR** register with the correct value. The value written to the **I2CMTPR** register represents the number of system clock periods in one SCL clock period. The TPR value is determined by the following equation:

```
TPR = (System Clock / (2 * (SCL_LP + SCL_HP) * SCL_CLK)) - 1;
TPR = (20MHz / (2 * (6 + 4) * 100000)) - 1;
TPR = 9
```

Write the **I2CMTPR** register with the value of 0x0000.0009.

6. Specify the slave address of the master and that the next operation will be a Send by writing the **I2CMSA** register with a value of 0x0000.0076. This sets the slave address to 0x3B.
7. Place data (byte) to be sent in the data register by writing the **I2CMDR** register with the desired data.
8. Initiate a single byte send of the data from Master to Slave by writing the **I2CMCS** register with a value of 0x0000.0007 (STOP, START, RUN).
9. Wait until the transmission completes by polling the **I2CMCS** register's BUSBSY bit until it has been cleared.

13.4 I^{2}C Register Map

Table 13-2 on page 303 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

Table 13-2. 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	305
0x004	I2CMCS	R/W	0x0000.0000	I2C Master Control/Status	306
0x008	I2CMDR	R/W	0x0000.0000	I2C Master Data	310
0x00C	I2CMTPR	R/W	0x0000.0001	I2C Master Timer Period	311
0x010	I2CMIMR	R/W	0x0000.0000	I2C Master Interrupt Mask	312
0x014	I2CMRIS	RO	0x0000.0000	I2C Master Raw Interrupt Status	313
0x018	I2CMMIS	RO	0x0000.0000	I2C Master Masked Interrupt Status	314
0x01C	I2CMICR	WO	0x0000.0000	I2C Master Interrupt Clear	315
0x020	I2CMCR	R/W	0x0000.0000	I2C Master Configuration	316
I^{2}C Slave					
0x000	I2CSOAR	R/W	0x0000.0000	I2C Slave Own Address	318
0x004	I2CSCSR	RO	0x0000.0000	I2C Slave Control/Status	319
0x008	I2CSDR	R/W	0x0000.0000	I2C Slave Data	321
0x00C	I2CSIMR	R/W	0x0000.0000	I2C Slave Interrupt Mask	322

Offset	Name	Type	Reset	Description	See page
0x010	I2CSRIS	RO	0x0000.0000	I2C Slave Raw Interrupt Status	323
0x014	I2CSMIS	RO	0x0000.0000	I2C Slave Masked Interrupt Status	324
0x018	I2CSICR	WO	0x0000.0000	I2C Slave Interrupt Clear	325

13.5 Register Descriptions (I2C Master)

The remainder of this section lists and describes the I2C master registers, in numerical order by address offset. See also “Register Descriptions (I2C Slave)” on page 317.

Register 1: I²C Master Slave Address (I2CMSA), offset 0x000

This register consists of eight bits: seven address bits (A6-A0), and a Receive/Send bit, which determines if the next operation is a Receive (High), or Send (Low).

I2C Master Slave Address (I2CMSA)

I2C Master 0 base: 0x4002.0000
Offset 0x000
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:1	SA	R/W	0	I²C Slave Address This field specifies bits A6 through A0 of the slave address.
0	R/S	R/W	0	Receive/Send The R/S bit specifies if the next operation is a Receive (High) or Send (Low). 0: Send 1: Receive

Register 2: I²C Master Control/Status (I2CMCS), offset 0x004

This register accesses four control bits when written, and accesses seven status bits when read.

The status register consists of seven bits, which when read determine the state of the I²C bus controller.

The control register consists of four bits: the `RUN`, `START`, `STOP`, and `ACK` bits. The `START` bit causes the generation of the START, or REPEATED START condition.

The `STOP` bit determines if the cycle stops at the end of the data cycle, or continues on to a burst. To generate a single send cycle, the **I²C Master Slave Address (I2CMSA)** register is written with the desired address, the `R/S` bit is set to 0, and the Control register is written with `ACK`=X (0 or 1), `STOP`=1, `START`=1, and `RUN`=1 to perform the operation and stop. When the operation is completed (or aborted due an error), the interrupt pin becomes active and the data may be read from the **I2CMDR** register. When the I²C module operates in Master receiver mode, the `ACK` bit must be set normally to logic 1. This causes the I²C bus controller to send an acknowledge automatically after each byte. This bit must be reset when the I²C bus controller requires no further data to be sent from the slave transmitter.

Read-Only Status Register

I2C Master Control/Status (I2CMCS)

I2C Master 0 base: 0x4002.0000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	BUSBSY	RO	0	Bus Busy This bit specifies the state of the I²C bus. If set, the bus is busy; otherwise, the bus is idle. The bit changes based on the START and STOP conditions.
5	IDLE	RO	0	I²C Idle This bit specifies the I²C controller state. If set, the controller is idle; otherwise the controller is not idle.
4	ARBLST	RO	0	Arbitration Lost This bit specifies the result of bus arbitration. If set, the controller lost arbitration; otherwise, the controller won arbitration.

Bit/Field	Name	Type	Reset	Description
3	DATACK	RO	0	Acknowledge Data This bit specifies the result of the last data operation. If set, the transmitted data was not acknowledged; otherwise, the data was acknowledged.
2	ADRACK	RO	0	Acknowledge Address This bit specifies the result of the last address operation. If set, the transmitted address was not acknowledged; otherwise, the address was acknowledged.
1	ERROR	RO	0	Error This bit specifies the result of the last bus operation. If set, an error occurred on the last operation; otherwise, no error was detected. The error can be from the slave address not being acknowledged, the transmit data not being acknowledged, or because the controller lost arbitration.
0	BUSY	RO	0	I^2C Busy This bit specifies the state of the controller. If set, the controller is busy; otherwise, the controller is idle. When the BUSY bit is set, the other status bits are not valid.

Write-Only Control Register

I2C Master Control/Status (I2CMCS)

I2C Master 0 base: 0x4002.0000
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ACK	WO	0	Data Acknowledge Enable When set, causes received data byte to be acknowledged automatically by the master. See field decoding in Table 13-3 on page 308.
2	STOP	WO	0	Generate STOP When set, causes the generation of the STOP condition. See field decoding in Table 13-3 on page 308.

Bit/Field	Name	Type	Reset	Description
1	START	WO	0	Generate START When set, causes the generation of a START or repeated START condition. See field decoding in Table 13-3 on page 308.
0	RUN	WO	0	I^2C Master Enable When set, allows the master to send or receive data. See field decoding in Table 13-3 on page 308.

Table 13-3. Write Field Decoding for I2CMCS[3:0] Field (Sheet 1 of 3)

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
	R/S	ACK	STOP	START	RUN	
Idle	0	X[a]	0	1	1	START condition followed by SEND (master goes to the Master Transmit state).
	0	X	1	1	1	START condition followed by a SEND and STOP condition (master remains in Idle state).
	1	0	0	1	1	START condition followed by RECEIVE operation with negative ACK (master goes to the Master Receive state).
	1	0	1	1	1	START condition followed by RECEIVE and STOP condition (master remains in Idle state).
	1	1	0	1	1	START condition followed by RECEIVE (master goes to the Master Receive state).
	1	1	1	1	1	Illegal.
	All other combinations not listed are non-operations.					NOP.
Master Transmit	X	X	0	0	1	SEND operation (master remains in Master Transmit state).
	X	X	1	0	0	STOP condition (master goes to Idle state).
	X	X	1	0	1	SEND followed by STOP condition (master goes to Idle state).
	0	X	0	1	1	Repeated START condition followed by a SEND (master remains in Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by SEND and STOP condition (master goes to Idle state).
	1	0	0	1	1	Repeated START condition followed by a RECEIVE operation with a negative ACK (master goes to Master Receive state).
	1	0	1	1	1	Repeated START condition followed by a SEND and STOP condition (master goes to Idle state).
	1	1	0	1	1	Repeated START condition followed by RECEIVE (master goes to Master Receive state).
	1	1	1	1	1	Illegal.
	All other combinations not listed are non-operations.					NOP.

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
	R/S	ACK	STOP	START	RUN	
Master Receive	X	0	0	0	1	RECEIVE operation with negative ACK (master remains in Master Receive state).
	X	X	1	0	0	STOP condition (master goes to Idle state).[b]
	X	0	1	0	1	RECEIVE followed by STOP condition (master goes to Idle state).
	X	1	0	0	1	RECEIVE operation (master remains in Master Receive state).
	X	1	1	0	1	Illegal.
	1	0	0	1	1	Repeated START condition followed by RECEIVE operation with a negative ACK (master remains in Master Receive state).
	1	0	1	1	1	Repeated START condition followed by RECEIVE and STOP condition (master goes to Idle state).
	1	1	0	1	1	Repeated START condition followed by RECEIVE (master remains in Master Receive state).
	0	X	0	1	1	Repeated START condition followed by SEND (master goes to Master Transmit state).
	0	X	1	1	1	Repeated START condition followed by SEND and STOP condition (master goes to Idle state).
	All other combinations not listed are non-operations.					NOP.

a. An X in a table cell indicates the bit can be 0 or 1.

b. In Master Receive mode, a STOP condition should be generated only after a Data Negative Acknowledge executed by the master or an Address Negative Acknowledge executed by the slave.

Register 3: I^2C Master Data (I2CMDR), offset 0x008

This register contains the data to be transmitted when in the Master Transmit state, and the data received when in the Master Receive state.

I2C Master Data (I2CMDR)

I2C Master 0 base: 0x4002.0000
Offset 0x008
Type R/W, reset 0x0000.0000

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

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

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

Register 4: I^{2}C Master Timer Period (I2CMTPR), offset 0x00C

This register specifies the period of the SCL clock.

I2C Master Timer Period (I2CMTPR)

I2C Master 0 base: 0x4002.0000
Offset 0x00C
Type R/W, reset 0x0000.0001

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

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TPR	R/W	0x1	SCL Clock Period This field specifies the period of the SCL clock. `SCL_PRD = 2*(1 + TPR)*(SCL_LP + SCL_HP)*CLK_PRD` where: `SCL_PRD` is the SCL line period (I^{2}C clock). `TPR` is the Timer Period register value (range of 1 to 255). `SCL_LP` is the SCL Low period (fixed at 6). `SCL_HP` is the SCL High period (fixed at 4).

Register 5: I²C Master Interrupt Mask (I2CMIMR), offset 0x010

This register controls whether a raw interrupt is promoted to a controller interrupt.

I2C Master Interrupt Mask (I2CMIMR)

I2C Master 0 base: 0x4002.0000
Offset 0x010
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IM	R/W	0	Interrupt Mask This bit controls whether a raw interrupt is promoted to a controller interrupt. If set, the interrupt is not masked and the interrupt is promoted; otherwise, the interrupt is masked.

Register 6: I^{2}C Master Raw Interrupt Status (I2CMRIS), offset 0x014

This register specifies whether an interrupt is pending.

I2C Master Raw Interrupt Status (I2CMRIS)

I2C Master 0 base: 0x4002.0000
Offset 0x014
Type RO, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	RIS	RO	0	Raw Interrupt Status This bit specifies the raw interrupt state (prior to masking) of the I^{2}C master block. If set, an interrupt is pending; otherwise, an interrupt is not pending.

Register 7: I^2C Master Masked Interrupt Status (I2CMMIS), offset 0x018

This register specifies whether an interrupt was signaled.

I2C Master Masked Interrupt Status (I2CMMIS)

I2C Master 0 base: 0x4002.0000
Offset 0x018
Type RO, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	MIS	RO	0	Masked Interrupt Status This bit specifies the raw interrupt state (after masking) of the I^2C master block. If set, an interrupt was signaled; otherwise, an interrupt has not been generated since the bit was last cleared.

Register 8: I^2C Master Interrupt Clear (I2CMICR), offset 0x01C

This register clears the raw interrupt.

I2C Master Interrupt Clear (I2CMICR)

I2C Master 0 base: 0x4002.0000
Offset 0x01C
Type WO, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IC	WO	0	Interrupt Clear This bit controls the clearing of the raw interrupt. A write of 1 clears the interrupt; otherwise, a write of 0 has no affect on the interrupt state. A read of this register returns no meaningful data.

Register 9: I^2C Master Configuration (I2CMCR), offset 0x020

This register configures the mode (Master or Slave) and sets the interface for test mode loopback.

I2C Master Configuration (I2CMCR)

I2C Master 0 base: 0x4002.0000
Offset 0x020
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SFE	R/W	0	I^2C Slave Function Enable This bit specifies whether the interface may operate in Slave mode. If set, Slave mode is enabled; otherwise, Slave mode is disabled.
4	MFE	R/W	0	I^2C Master Function Enable This bit specifies whether the interface may operate in Master mode. If set, Master mode is enabled; otherwise, Master mode is disabled and the interface clock is disabled.
3:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	LPBK	R/W	0	I^2C Loopback This bit specifies whether the interface is operating normally or in Loopback mode. If set, the device is put in a test mode loopback configuration; otherwise, the device operates normally.

13.6 Register Descriptions (I2C Slave)

The remainder of this section lists and describes the I^2C slave registers, in numerical order by address offset. See also “Register Descriptions (I^2C Master)” on page 304.

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

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

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

Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	OAR	R/W	0x00	I^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
Offset 0x004
Type RO, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	FBR	RO	0	First Byte Received Indicates that the first byte following the slave's own address is received. This bit is only valid when the `RREQ` bit is set, and is automatically cleared when data has been read from the **I2CSDR** register. **Note:** This bit is not used for slave transmit operations.
1	TREQ	RO	0	Transmit Request This bit specifies the state of the I²C slave with regards to outstanding transmit requests. If set, the I²C unit has been addressed as a slave transmitter and uses clock stretching to delay the master until data has been written to the **I2CSDR** register. Otherwise, there is no outstanding transmit request.
0	RREQ	RO	0	Receive Request This bit specifies the status of the I²C slave with regards to outstanding receive requests. If set, the I²C unit has outstanding receive data from the I²C master and uses clock stretching to delay the master until the data has been read from the **I2CSDR** register. Otherwise, no receive data is outstanding.

Write-Only Control Register

I2C Slave Control/Status (I2CSCSR)

I2C Slave 0 base: 0x4002.0800
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	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	DA	WO	0	Device Active 1=Enables the I^{2}C slave operation. 0=Disables the I^{2}C slave operation.

Register 12: I^{2}C Slave Data (I2CSDR), offset 0x008

This register contains the data to be transmitted when in the Slave Transmit state, and the data received when in the Slave Receive state.

I2C Slave Data (I2CSDR)

I2C Slave 0 base: 0x4002.0800
Offset 0x008
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x0	Data for Transfer This field contains the data for transfer during a slave receive or transmit operation.

Register 13: I^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
Offset 0x00C
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IM	R/W	0	Interrupt Mask This bit controls whether a raw interrupt is promoted to a controller interrupt. If set, the interrupt is not masked and the interrupt is promoted; otherwise, the interrupt is masked.

Register 14: I²C Slave Raw Interrupt Status (I2CSRIS), offset 0x010

This register specifies whether an interrupt is pending.

I2C Slave Raw Interrupt Status (I2CSRIS)

I2C Slave 0 base: 0x4002.0800
Offset 0x010
Type RO, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	RIS	RO	0	Raw Interrupt Status This bit specifies the raw interrupt state (prior to masking) of the I²C slave block. If set, an interrupt is pending; otherwise, an interrupt is not pending.

Register 15: I²C Slave Masked Interrupt Status (I2CSMIS), offset 0x014

This register specifies whether an interrupt was signaled.

I2C Slave Masked Interrupt Status (I2CSMIS)

I2C Slave 0 base: 0x4002.0800
Offset 0x014
Type RO, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	MIS	RO	0	Masked Interrupt Status This bit specifies the raw interrupt state (after masking) of the I²C slave block. If set, an interrupt was signaled; otherwise, an interrupt has not been generated since the bit was last cleared.

Register 16: I²C Slave Interrupt Clear (I2CSICR), offset 0x018

This register clears the raw interrupt.

I2C Slave Interrupt Clear (I2CSICR)

I2C Slave 0 base: 0x4002.0800
Offset 0x018
Type WO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved															IC
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	0x00	Software should not rely on the value of 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	Clear Interrupt This bit controls the clearing of the raw interrupt. A write of 1 clears the interrupt; otherwise a write of 0 has no affect on the interrupt state. A read of this register returns no meaningful data.

14 Analog Comparators

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

The LM3S600 controller provides three independent integrated analog comparators that can be configured to drive an output or generate an interrupt.

Note: Not all comparators have the option to drive an output pin. See the Comparator Operating Mode tables for more information.

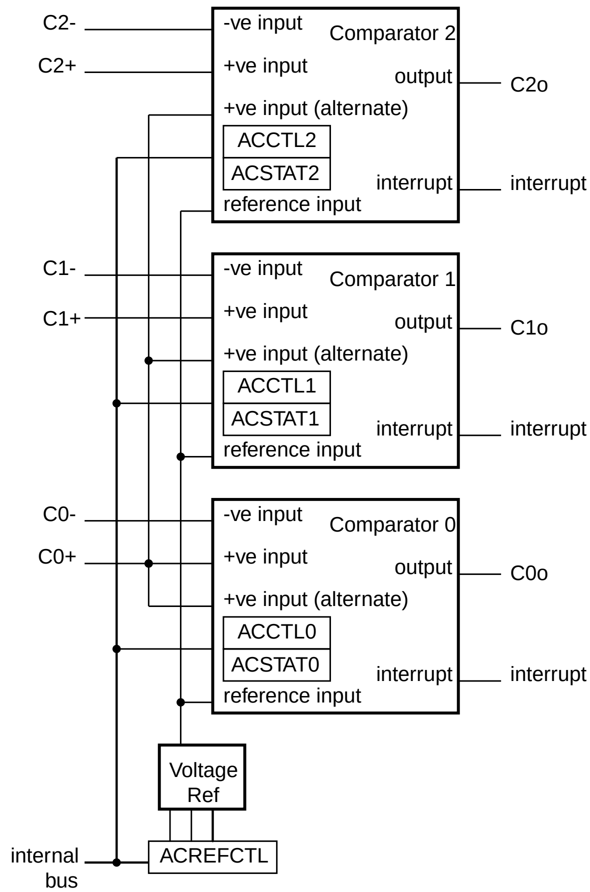
A comparator can compare a test voltage against any one of these voltages:

- An individual external reference voltage
- A shared single external reference voltage
- A shared internal reference voltage

The comparator can provide its output to a device pin, acting as a replacement for an analog comparator on the board, or it can be used to signal the application via interrupts to cause it to start capturing a sample sequence.

14.1 Block Diagram

Figure 14-1. Analog Comparator Module Block Diagram



14.2 Functional Description

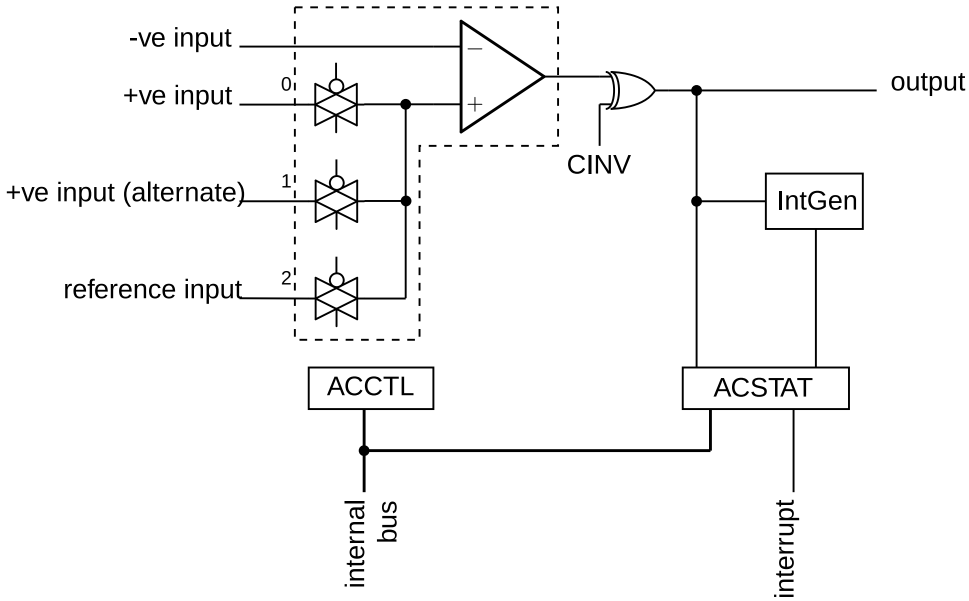
Important: It is recommended that the Digital-Input enable (the `GPIODEN` bit in the GPIO module) for the analog input pin be disabled to prevent excessive current draw from the I/O pads.

The comparator compares the VIN- and VIN+ inputs to produce an output, VOUT.

```
VIN- < VIN+, VOUT = 1
VIN- > VIN+, VOUT = 0
```

As shown in Figure 14-2 on page 328, the input source for VIN- is an external input. In addition to an external input, input sources for VIN+ can be the +ve input of comparator 0 or an internal reference.

Figure 14-2. Structure of Comparator Unit



A comparator is configured through two status/control registers (**ACCTL** and **ACSTAT**). The internal reference is configured through one control register (**ACREFCTL**). Interrupt status and control is configured through three registers (**ACMIS**, **ACRIS**, and **ACINTEN**). The operating modes of the comparators are shown in the Comparator Operating Mode tables.

Typically, the comparator output is used internally to generate controller interrupts. It may also be used to drive an external pin.

Important: Certain register bit values must be set before using the analog comparators. The proper pad configuration for the comparator input and output pins are described in the Comparator Operating Mode tables.

Table 14-1. Comparator 0 Operating Modes

ACCNTL0	Comparator 0			
ASRCP	**VIN-**	**VIN+**	**Output**	**Interrupt**
00	C0-	C0+	C0o	yes
01	C0-	C0+	C0o	yes
10	C0-	Vref	C0o	yes
11	C0-	reserved	C0o	yes

Table 14-2. Comparator 1 Operating Modes

ACCNTL1	Comparator 1			
ASRCP	**VIN-**	**VIN+**	**Output**	**Interrupt**
00	C1-	C1o/C1+[a]	C1o/C1+	yes
01	C1-	C0+	C1o/C1+	yes
10	C1-	Vref	C1o/C1+	yes
11	C1-	reserved	C1o/C1+	yes

a. C1o and C1+ signals share a single pin and may only be used as one or the other.

Table 14-3. Comparator 2 Operating Modes

ACCNTL2	Comparator 2			
ASRCP	**VIN-**	**VIN+**	**Output**	**Interrupt**
00	C2-	C2o/C2+[a]	C2o/C2+	yes
01	C2-	C0+	C2o/C2+	yes
10	C2-	Vref	C2o/C2+	yes
11	C2-	reserved	C2o/C2+	yes

a. C2o and C2+ signals share a single pin and may only be used as one or the other.

14.2.1 Internal Reference Programming

The structure of the internal reference is shown in Figure 14-3 on page 329. This is controlled by a single configuration register (**ACREFCTL**). Table 14-4 on page 329 shows the programming options to develop specific internal reference values, to compare an external voltage against a particular voltage generated internally.

Figure 14-3. Comparator Internal Reference Structure

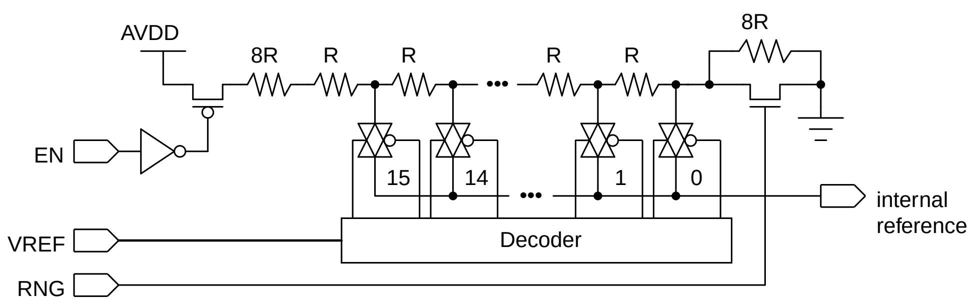


Table 14-4. Internal Reference Voltage and ACREFCTL Field Values

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.

ACREFCTL Register		Output Reference Voltage Based on VREF Field Value
EN Bit Value	**RNG Bit Value**	
EN=1	RNG=0	Total resistance in ladder is 32 R. $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF + 8)}{32}$ $V_{REF} = 0.825 + 0.103 \ VREF$ The range of internal reference in this mode is 0.825-2.37 V.
	RNG=1	Total resistance in ladder is 24 R. $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF)}{24}$ $V_{REF} = 0.1375 \times V_{REF}$ The range of internal reference for this mode is 0.0-2.0625 V.

14.3 Initialization and Configuration

The following example shows how to configure an analog comparator to read back its output value from an internal register.

1. Enable the analog comparator 0 clock by writing a value of 0x0010.0000 to the **RCGC1** register in the System Control module.
2. In the GPIO module, enable the GPIO port/pin associated with `C0-` as a GPIO input.
3. Configure the internal voltage reference to 1.65 V by writing the **ACREFCTL** register with the value 0x0000.030C.
4. Configure comparator 0 to use the internal voltage reference and to *not* invert the output on the `C0o` pin by writing the **ACCTL0** register with the value of 0x0000.040C.
5. Delay for some time.
6. Read the comparator output value by reading the **ACSTAT0** register's `OVAL` value.

Change the level of the signal input on `C0-` to see the `OVAL` value change.

14.4 Register Map

Table 14-5 on page 331 lists the comparator registers. The offset listed is a hexadecimal increment to the register's address, relative to the Analog Comparator base address of 0x4003.C000.

Table 14-5. Analog Comparators Register Map

Offset	Name	Type	Reset	Description	See page
0x00	ACMIS	R/W1C	0x0000.0000	Analog Comparator Masked Interrupt Status	332
0x04	ACRIS	RO	0x0000.0000	Analog Comparator Raw Interrupt Status	333
0x08	ACINTEN	R/W	0x0000.0000	Analog Comparator Interrupt Enable	334
0x10	ACREFCTL	R/W	0x0000.0000	Analog Comparator Reference Voltage Control	335
0x20	ACSTAT0	RO	0x0000.0000	Analog Comparator Status 0	336
0x24	ACCTL0	R/W	0x0000.0000	Analog Comparator Control 0	337
0x40	ACSTAT1	RO	0x0000.0000	Analog Comparator Status 1	336
0x44	ACCTL1	R/W	0x0000.0000	Analog Comparator Control 1	337
0x60	ACSTAT2	RO	0x0000.0000	Analog Comparator Status 2	336
0x64	ACCTL2	R/W	0x0000.0000	Analog Comparator Control 2	337

14.5 Register Descriptions

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

Register 1: Analog Comparator Masked Interrupt Status (ACMIS), offset 0x00

This register provides a summary of the interrupt status (masked) of the comparator.

Analog Comparator Masked Interrupt Status (ACMIS)

Base 0x4003.C000
Offset 0x00
Type R/W1C, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													IN2	IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	IN2	R/W1C	0	Comparator 2 Masked Interrupt Status Gives the masked interrupt state of this interrupt. Write 1 to this bit to clear the pending interrupt.
1	IN1	R/W1C	0	Comparator 1 Masked Interrupt Status Gives the masked interrupt state of this interrupt. Write 1 to this bit to clear the pending interrupt.
0	IN0	R/W1C	0	Comparator 0 Masked Interrupt Status Gives the masked interrupt state of this interrupt. Write 1 to this bit to clear the pending interrupt.

Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x04

This register provides a summary of the interrupt status (raw) of the comparator.

Analog Comparator Raw Interrupt Status (ACRIS)

Base 0x4003.C000
Offset 0x04
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													IN2	IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	IN2	RO	0	Comparator 2 Interrupt Status When set, indicates that an interrupt has been generated by comparator 2.
1	IN1	RO	0	Comparator 1 Interrupt Status When set, indicates that an interrupt has been generated by comparator 1.
0	IN0	RO	0	Comparator 0 Interrupt Status When set, indicates that an interrupt has been generated by comparator 0.

Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x08

This register provides the interrupt enable for the comparator.

Analog Comparator Interrupt Enable (ACINTEN)

Base 0x4003.C000
Offset 0x08
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													IN2	IN1	IN0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	IN2	R/W	0	Comparator 2 Interrupt Enable When set, enables the controller interrupt from the comparator 2 output
1	IN1	R/W	0	Comparator 1 Interrupt Enable When set, enables the controller interrupt from the comparator 1 output.
0	IN0	R/W	0	Comparator 0 Interrupt Enable When set, enables the controller interrupt from the comparator 0 output.

Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x10

This register specifies whether the resistor ladder is powered on as well as the range and tap.

Analog Comparator Reference Voltage Control (ACREFCTL)

Base 0x4003.C000
Offset 0x10
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9	EN	R/W	0	Resistor Ladder Enable The EN bit specifies whether the resistor ladder is powered on. If 0, the resistor ladder is unpowered. If 1, the resistor ladder is connected to the analog V_{DD}. This bit is reset to 0 so that the internal reference consumes the least amount of power if not used and programmed.
8	RNG	R/W	0	Resistor Ladder Range The RNG bit specifies the range of the resistor ladder. If 0, the resistor ladder has a total resistance of 32 R. If 1, the resistor ladder has a total resistance of 24 R.
7:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	VREF	R/W	0x00	Resistor Ladder Voltage Ref The 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 14-4 on page 329 for some output reference voltage examples.

Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x20

Register 6: Analog Comparator Status 1 (ACSTAT1), offset 0x40

Register 7: Analog Comparator Status 2 (ACSTAT2), offset 0x60

These registers specify the current output value of the comparator.

Analog Comparator Status 0 (ACSTAT0)

Base 0x4003.C000
Offset 0x20
Type RO, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved														OVAL	reserved
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
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	OVAL	RO	0	Comparator Output Value The OVAL bit specifies the current output value of the comparator.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 8: Analog Comparator Control 0 (ACCTL0), offset 0x24

Register 9: Analog Comparator Control 1 (ACCTL1), offset 0x44

Register 10: Analog Comparator Control 2 (ACCTL2), offset 0x64

These registers configure the comparator's input and output.

Analog Comparator Control 0 (ACCTL0)

Base 0x4003.C000
Offset 0x24
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved					ASRCP		reserved				ISLVAL	ISEN		CINV	reserved
Type	RO	RO	RO	RO	RO	R/W	R/W	RO	RO	RO	RO	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

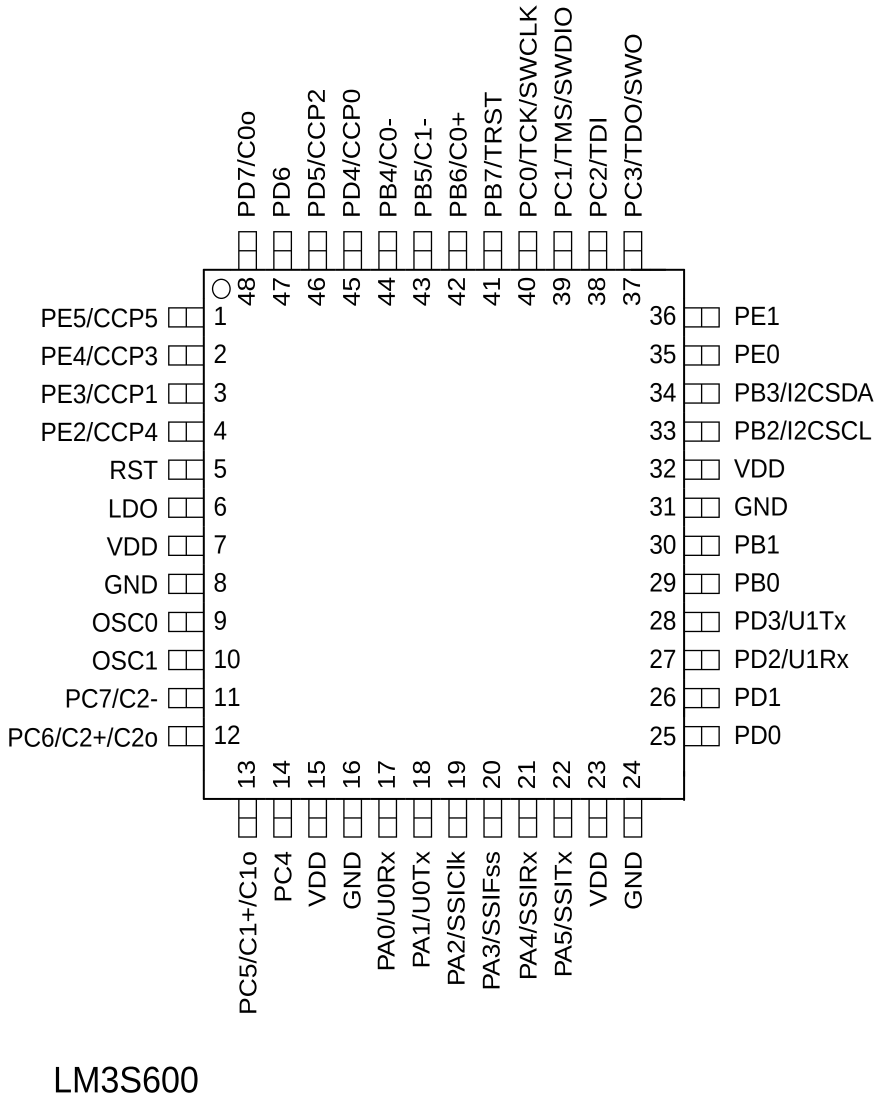
Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10:9	ASRCP	R/W	0x00	Analog Source Positive The ASRCP field specifies the source of input voltage to the VIN+ terminal of the comparator. The encodings for this field are as follows: Value Function 0x0 Pin value 0x1 Pin value of C0+ 0x2 Internal voltage reference 0x3 Reserved
8:5	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	ISLVAL	R/W	0	Interrupt Sense Level Value The ISLVAL bit specifies the sense value of the input that generates an interrupt if in Level Sense mode. If 0, an interrupt is generated if the comparator output is Low. Otherwise, an interrupt is generated if the comparator output is High.

Bit/Field	Name	Type	Reset	Description
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 Function 0x0 Level sense, see ISLVAL 0x1 Falling edge 0x2 Rising edge 0x3 Either edge
1	CINV	R/W	0	Comparator Output Invert The CINV bit conditionally inverts the output of the comparator. If 0, the output of the comparator is unchanged. If 1, the output of the comparator is inverted prior to being processed by hardware.
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.

15 Pin Diagram

Figure 15-1 on page 339 shows the pin diagram and pin-to-signal-name mapping.

Figure 15-1. Pin Connection Diagram



16 Signal Tables

The following tables list the signals available for each pin. Functionality is enabled by software with the **GPIOAFSEL** register.

Important: All multiplexed pins are GPIOs by default, with the exception of the five JTAG pins (PB7 and PC[3:0]) which default to the JTAG functionality.

Table 16-1 on page 340 shows the pin-to-signal-name mapping, including functional characteristics of the signals. Table 16-2 on page 342 lists the signals in alphabetical order by signal name.

Table 16-3 on page 344 groups the signals by functionality, except for GPIOs. Table 16-4 on page 345 lists the GPIO pins and their alternate functionality.

Table 16-1. Signals by Pin Number

Pin Number	Pin Name	Pin Type	Buffer Type	Description
1	PE5	I/O	TTL	GPIO port E bit 5
	CCP5	I/O	TTL	Capture/Compare/PWM 5
2	PE4	I/O	TTL	GPIO port E bit 4
	CCP3	I/O	TTL	Capture/Compare/PWM 3
3	PE3	I/O	TTL	GPIO port E bit 3
	CCP1	I/O	TTL	Capture/Compare/PWM 1
4	PE2	I/O	TTL	GPIO port E bit 2
	CCP4	I/O	TTL	Capture/Compare/PWM 4
5	$\overline{RST}$	I	TTL	System reset input.
6	LDO	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater.
7	VDD	-	Power	Positive supply for I/O and some logic.
8	GND	-	Power	Ground reference for logic and I/O pins.
9	OSC0	I	Analog	Main oscillator crystal input or an external clock reference input.
10	OSC1	O	Analog	Main oscillator crystal output.
11	PC7	I/O	TTL	GPIO port C bit 7
	C2-	I	Analog	Analog comparator 2 negative input
12	PC6	I/O	TTL	GPIO port C bit 6
	C2+	I	Analog	Analog comparator positive input
	C2o	O	TTL	Analog comparator 2 output
13	PC5	I/O	TTL	GPIO port C bit 5
	C1+	I	Analog	Analog comparator positive input
	C1o	O	TTL	Analog comparator 1 output
14	PC4	I/O	TTL	GPIO port C bit 4
15	VDD	-	Power	Positive supply for I/O and some logic.
16	GND	-	Power	Ground reference for logic and I/O pins.
17	PA0	I/O	TTL	GPIO port A bit 0
	U0Rx	I	TTL	UART module 0 receive

Pin Number	Pin Name	Pin Type	Buffer Type	Description
18	PA1	I/O	TTL	GPIO port A bit 1
	U0Tx	O	TTL	UART module 0 transmit
19	PA2	I/O	TTL	GPIO port A bit 2
	SSIClk	I/O	TTL	SSI clock
20	PA3	I/O	TTL	GPIO port A bit 3
	SSIFss	I/O	TTL	SSI frame
21	PA4	I/O	TTL	GPIO port A bit 4
	SSIRx	I	TTL	SSI module 0 receive
22	PA5	I/O	TTL	GPIO port A bit 5
	SSITx	O	TTL	SSI module 0 transmit
23	VDD	-	Power	Positive supply for I/O and some logic.
24	GND	-	Power	Ground reference for logic and I/O pins.
25	PD0	I/O	TTL	GPIO port D bit 0
26	PD1	I/O	TTL	GPIO port D bit 1
27	PD2	I/O	TTL	GPIO port D bit 2
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
28	PD3	I/O	TTL	GPIO port D bit 3
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
29	PB0	I/O	TTL	GPIO port B bit 0
30	PB1	I/O	TTL	GPIO port B bit 1
31	GND	-	Power	Ground reference for logic and I/O pins.
32	VDD	-	Power	Positive supply for I/O and some logic.
33	PB2	I/O	TTL	GPIO port B bit 2
	I2CSCL	I/O	OD	I2C module 0 clock
34	PB3	I/O	TTL	GPIO port B bit 3
	I2CSDA	I/O	OD	I2C module 0 data
35	PE0	I/O	TTL	GPIO port E bit 0
36	PE1	I/O	TTL	GPIO port E bit 1
37	PC3	I/O	TTL	GPIO port C bit 3
	TDO	O	TTL	JTAG TDO and SWO
	SWO	O	TTL	JTAG TDO and SWO
38	PC2	I/O	TTL	GPIO port C bit 2
	TDI	I	TTL	JTAG TDI
39	PC1	I/O	TTL	GPIO port C bit 1
	TMS	I/O	TTL	JTAG TMS and SWDIO
	SWDIO	I/O	TTL	JTAG TMS and SWDIO
40	PC0	I/O	TTL	GPIO port C bit 0
	TCK	I	TTL	JTAG/SWD CLK
	SWCLK	I	TTL	JTAG/SWD CLK
41	PB7	I/O	TTL	GPIO port B bit 7
	TRST	I	TTL	JTAG TRSTn

Pin Number	Pin Name	Pin Type	Buffer Type	Description
42	PB6	I/O	TTL	GPIO port B bit 6
	C0+	I	Analog	Analog comparator 0 positive input
43	PB5	I/O	TTL	GPIO port B bit 5
	C1-	I	Analog	Analog comparator 1 negative input
44	PB4	I/O	TTL	GPIO port B bit 4
	C0-	I	Analog	Analog comparator 0 negative input
45	PD4	I/O	TTL	GPIO port D bit 4
	CCP0	I/O	TTL	Capture/Compare/PWM 0
46	PD5	I/O	TTL	GPIO port D bit 5
	CCP2	I/O	TTL	Capture/Compare/PWM 2
47	PD6	I/O	TTL	GPIO port D bit 6
48	PD7	I/O	TTL	GPIO port D bit 7
	C0o	O	TTL	Analog comparator 0 output

Table 16-2. Signals by Signal Name

Pin Name	Pin Number	Pin Type	Buffer Type	Description
C0+	42	I	Analog	Analog comparator 0 positive input
C0-	44	I	Analog	Analog comparator 0 negative input
C0o	48	O	TTL	Analog comparator 0 output
C1+	13	I	Analog	Analog comparator positive input
C1-	43	I	Analog	Analog comparator 1 negative input
C1o	13	O	TTL	Analog comparator 1 output
C2+	12	I	Analog	Analog comparator positive input
C2-	11	I	Analog	Analog comparator 2 negative input
C2o	12	O	TTL	Analog comparator 2 output
CCP0	45	I/O	TTL	Capture/Compare/PWM 0
CCP1	3	I/O	TTL	Capture/Compare/PWM 1
CCP2	46	I/O	TTL	Capture/Compare/PWM 2
CCP3	2	I/O	TTL	Capture/Compare/PWM 3
CCP4	4	I/O	TTL	Capture/Compare/PWM 4
CCP5	1	I/O	TTL	Capture/Compare/PWM 5
GND	8	-	Power	Ground reference for logic and I/O pins.
GND	16	-	Power	Ground reference for logic and I/O pins.
GND	24	-	Power	Ground reference for logic and I/O pins.
GND	31	-	Power	Ground reference for logic and I/O pins.
I2CSCL	33	I/O	OD	I2C module 0 clock
I2CSDA	34	I/O	OD	I2C module 0 data
LDO	6	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater.
OSC0	9	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	10	O	Analog	Main oscillator crystal output.

Pin Name	Pin Number	Pin Type	Buffer Type	Description
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
PB0	29	I/O	TTL	GPIO port B bit 0
PB1	30	I/O	TTL	GPIO port B bit 1
PB2	33	I/O	TTL	GPIO port B bit 2
PB3	34	I/O	TTL	GPIO port B bit 3
PB4	44	I/O	TTL	GPIO port B bit 4
PB5	43	I/O	TTL	GPIO port B bit 5
PB6	42	I/O	TTL	GPIO port B bit 6
PB7	41	I/O	TTL	GPIO port B bit 7
PC0	40	I/O	TTL	GPIO port C bit 0
PC1	39	I/O	TTL	GPIO port C bit 1
PC2	38	I/O	TTL	GPIO port C bit 2
PC3	37	I/O	TTL	GPIO port C bit 3
PC4	14	I/O	TTL	GPIO port C bit 4
PC5	13	I/O	TTL	GPIO port C bit 5
PC6	12	I/O	TTL	GPIO port C bit 6
PC7	11	I/O	TTL	GPIO port C bit 7
PD0	25	I/O	TTL	GPIO port D bit 0
PD1	26	I/O	TTL	GPIO port D bit 1
PD2	27	I/O	TTL	GPIO port D bit 2
PD3	28	I/O	TTL	GPIO port D bit 3
PD4	45	I/O	TTL	GPIO port D bit 4
PD5	46	I/O	TTL	GPIO port D bit 5
PD6	47	I/O	TTL	GPIO port D bit 6
PD7	48	I/O	TTL	GPIO port D bit 7
PE0	35	I/O	TTL	GPIO port E bit 0
PE1	36	I/O	TTL	GPIO port E bit 1
PE2	4	I/O	TTL	GPIO port E bit 2
PE3	3	I/O	TTL	GPIO port E bit 3
PE4	2	I/O	TTL	GPIO port E bit 4
PE5	1	I/O	TTL	GPIO port E bit 5
$\overline{RST}$	5	I	TTL	System reset input.
SSIClk	19	I/O	TTL	SSI clock
SSIFss	20	I/O	TTL	SSI frame
SSIRx	21	I	TTL	SSI module 0 receive
SSITx	22	O	TTL	SSI module 0 transmit
SWCLK	40	I	TTL	JTAG/SWD CLK

Pin Name	Pin Number	Pin Type	Buffer Type	Description
SWDIO	39	I/O	TTL	JTAG TMS and SWDIO
SWO	37	O	TTL	JTAG TDO and SWO
TCK	40	I	TTL	JTAG/SWD CLK
TDI	38	I	TTL	JTAG TDI
TDO	37	O	TTL	JTAG TDO and SWO
TMS	39	I/O	TTL	JTAG TMS and SWDIO
TRST	41	I	TTL	JTAG TRSTn
U0Rx	17	I	TTL	UART module 0 receive
U0Tx	18	O	TTL	UART module 0 transmit
U1Rx	27	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
U1Tx	28	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
VDD	7	-	Power	Positive supply for I/O and some logic.
VDD	15	-	Power	Positive supply for I/O and some logic.
VDD	23	-	Power	Positive supply for I/O and some logic.
VDD	32	-	Power	Positive supply for I/O and some logic.

Table 16-3. Signals by Function, Except for GPIO

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
Analog Comparators	C0+	42	I	Analog	Analog comparator 0 positive input
	C0-	44	I	Analog	Analog comparator 0 negative input
	C0o	48	O	TTL	Analog comparator 0 output
	C1+	13	I	Analog	Analog comparator positive input
	C1-	43	I	Analog	Analog comparator 1 negative input
	C1o	13	O	TTL	Analog comparator 1 output
	C2+	12	I	Analog	Analog comparator positive input
	C2-	11	I	Analog	Analog comparator 2 negative input
	C2o	12	O	TTL	Analog comparator 2 output
General-Purpose Timers	CCP0	45	I/O	TTL	Capture/Compare/PWM 0
	CCP1	3	I/O	TTL	Capture/Compare/PWM 1
	CCP2	46	I/O	TTL	Capture/Compare/PWM 2
	CCP3	2	I/O	TTL	Capture/Compare/PWM 3
	CCP4	4	I/O	TTL	Capture/Compare/PWM 4
	CCP5	1	I/O	TTL	Capture/Compare/PWM 5
I2C	I2CSCL	33	I/O	OD	I2C module 0 clock
	I2CSDA	34	I/O	OD	I2C module 0 data

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
JTAG/SWD/SWO	SWCLK	40	I	TTL	JTAG/SWD CLK
	SWDIO	39	I/O	TTL	JTAG TMS and SWDIO
	SWO	37	O	TTL	JTAG TDO and SWO
	TCK	40	I	TTL	JTAG/SWD CLK
	TDI	38	I	TTL	JTAG TDI
	TDO	37	O	TTL	JTAG TDO and SWO
	TMS	39	I/O	TTL	JTAG TMS and SWDIO
Power	GND	8	-	Power	Ground reference for logic and I/O pins.
	GND	16	-	Power	Ground reference for logic and I/O pins.
	GND	24	-	Power	Ground reference for logic and I/O pins.
	GND	31	-	Power	Ground reference for logic and I/O pins.
	LDO	6	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater.
	VDD	7	-	Power	Positive supply for I/O and some logic.
	VDD	15	-	Power	Positive supply for I/O and some logic.
	VDD	23	-	Power	Positive supply for I/O and some logic.
	VDD	32	-	Power	Positive supply for I/O and some logic.
SSI	SSIClk	19	I/O	TTL	SSI clock
	SSIFss	20	I/O	TTL	SSI frame
	SSIRx	21	I	TTL	SSI module 0 receive
	SSITx	22	O	TTL	SSI module 0 transmit
System Control & Clocks	OSC0	9	I	Analog	Main oscillator crystal input or an external clock reference input.
	OSC1	10	O	Analog	Main oscillator crystal output.
	RST	5	I	TTL	System reset input.
	TRST	41	I	TTL	JTAG TRSTn
UART	U0Rx	17	I	TTL	UART module 0 receive
	U0Tx	18	O	TTL	UART module 0 transmit
	U1Rx	27	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U1Tx	28	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.

Table 16-4. GPIO Pins and Alternate Functions

GPIO Pin	Pin Number	Multiplexed Function	Multiplexed Function
PA0	17	U0Rx	
PA1	18	U0Tx	
PA2	19	SSIClk	
PA3	20	SSIFss	
PA4	21	SSIRx	
PA5	22	SSITx	
PB0	29		

GPIO Pin	Pin Number	Multiplexed Function	Multiplexed Function
PB1	30		
PB2	33	I2CSCL	
PB3	34	I2CSDA	
PB4	44	C0-	
PB5	43	C1-	
PB6	42	C0+	
PB7	41	$\overline{\text{TRST}}$	
PC0	40	TCK	SWCLK
PC1	39	TMS	SWDIO
PC2	38	TDI	
PC3	37	TDO	SWO
PC4	14		
PC5	13	C1+	C1o
PC6	12	C2+	C2o
PC7	11	C2-	
PD0	25		
PD1	26		
PD2	27	U1Rx	
PD3	28	U1Tx	
PD4	45	CCP0	
PD5	46	CCP2	
PD6	47		
PD7	48	C0o	
PE0	35		
PE1	36		
PE2	4	CCP4	
PE3	3	CCP1	
PE4	2	CCP3	
PE5	1	CCP5	

17 Operating Characteristics

Table 17-1. Temperature Characteristics

Characteristic	Symbol	Value	Unit
Operating temperature range[a]	T_A	-40 to +85	°C

a. Maximum storage temperature is 150°C.

Table 17-2. Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance (junction to ambient)[a]	Θ_{JA}	76	°C/W
Average junction temperature[b]	T_J	$T_A + (P_{AVG} \cdot \Theta_{JA})$	°C
Maximum junction temperature	T_{JMAX}	115 [c]	°C

a. Junction to ambient thermal resistance θ_{JA} numbers are determined by a package simulator.

b. Power dissipation is a function of temperature.

c. T_{JMAX} calculation is based on power consumption values and conditions as specified in "Power Specifications" on page 383 of the data sheet.

18 Electrical Characteristics

18.1 DC Characteristics

18.1.1 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 18-1. Maximum Ratings

Characteristic[a]	Symbol	Value	Unit
Supply voltage range (V_{DD})	V_{DD}	0.0 to +3.6	V
Input voltage	V_{IN}	-0.3 to 5.5	V
Maximum current for pins, excluding pins operating as GPIOs	I	100	mA
Maximum current for GPIO pins	I	100	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}).

18.1.2 Recommended DC Operating Conditions

Table 18-2. Recommended DC Operating Conditions

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{DD}	Supply voltage	3.0	3.3	3.6	V
V_{IH}	High-level input voltage	2.0	-	5.0	V
V_{IL}	Low-level input voltage	-0.3	-	1.3	V
V_{SIH}	High-level input voltage for Schmitt trigger inputs	0.8 * V_{DD}	-	V_{DD}	V
V_{SIL}	Low-level input voltage for Schmitt trigger inputs	0	-	0.2 * V_{DD}	V
V_{OH}	High-level output voltage	2.4	-	-	V
V_{OL}	Low-level output voltage	-	-	0.4	V
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

18.1.3 On-Chip Low Drop-Out (LDO) Regulator Characteristics

Table 18-3. LDO Regulator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{LDOOUT}	Programmable internal (logic) power supply output value	2.25		2.75	V
	Output voltage accuracy	-	2%	-	%
t_{PON}	Power-on time	-	-	100	µs
t_{ON}	Time on	-	-	200	µs
t_{OFF}	Time off	-	-	100	µs
V_{STEP}	Step programming incremental voltage	-	50	-	mV
C_{LDO}	External filter capacitor size for internal power supply	1.0	-	3.0	µF

18.1.4 Power Specifications

The power measurements specified in the tables that follow are run on the core processor using SRAM with the following specifications (except as noted):

- V_{DD} = 3.3 V
- Temperature = 25°C

Table 18-4. Detailed Power Specifications

Parameter	Parameter Name	Conditions	Nom	Max	Unit
I_{DD_RUN}	Run mode 1 (Flash loop)	LDO = 2.50 V Code = while(1){} executed in Flash Peripherals = All clock-gated ON System Clock = 50 MHz (with PLL)	95	110	mA
	Run mode 2 (Flash loop)	LDO = 2.50 V Code = while(1){} executed in Flash Peripherals = All clock-gated OFF System Clock = 50 MHz (with PLL)	60	75	mA
	Run mode 1 (SRAM loop)	LDO = 2.50 V Code = while(1){} executed in SRAM Peripherals = All clock-gated ON System Clock = 50 MHz (with PLL)	85	95	mA
	Run mode 2 (SRAM loop)	LDO = 2.50 V Code = while(1){} executed in SRAM Peripherals = All clock-gated OFF System Clock = 50 MHz (with PLL)	50	60	mA
I_{DD_SLEEP}	Sleep mode	LDO = 2.50 V Peripherals = All clock-gated OFF System Clock = 50 MHz (with PLL)	19	22	mA

Parameter	Parameter Name	Conditions	Nom	Max	Unit
$I_{DD\_DEEPSLEEP}$	Deep-Sleep mode	LDO = 2.25 V Peripherals = All OFF System Clock = MOSC/16	950	1150	µA

18.1.5 Flash Memory Characteristics

Table 18-5. Flash Memory Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
PE_{CYC}	Number of guaranteed program/erase cycles before failure[a]	1000	-	-	cycles
T_{RET}	Data retention at average operating temperature of 85˚C	10	-	-	years
T_{PROG}	Word program time	20	-	-	µs
T_{ERASE}	Page erase time	20	-	-	ms
T_{ME}	Mass erase time	200	-	-	ms

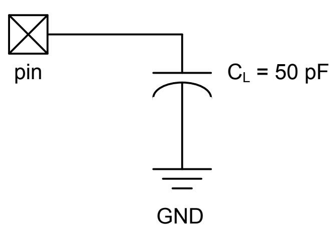
a. A program/erase cycle is defined as switching the bits from 1-> 0 -> 1.

18.2 AC Characteristics

18.2.1 Load Conditions

Unless otherwise specified, the following conditions are true for all timing measurements. Timing measurements are for 4-mA drive strength.

Figure 18-1. Load Conditions



18.2.2 Clocks

Table 18-6. Phase Locked Loop (PLL) Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$f_{ref\_crystal}$	Crystal reference[a]	3.579545	-	8.192	MHz
f_{ref_ext}	External clock reference[a]	3.579545	-	8.192	MHz
f_{pll}	PLL frequency[b]	-	200	-	MHz
T_{READY}	PLL lock time	-	-	0.5	ms

a. The exact value is determined by the crystal value programmed into the XTAL field of the **Run-Mode Clock Configuration (RCC)** register.

b. PLL frequency is automatically calculated by the hardware based on the XTAL field of the **RCC** register.

Table 18-7. Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
f_{IOSC}	Internal oscillator frequency	7	12	22	MHz

Parameter	Parameter Name	Min	Nom	Max	Unit
f_{MOSC}	Main oscillator frequency	1	-	8	MHz
t_{MOSC_per}	Main oscillator period	125	-	1000	ns
$f_{ref\_crystal\_bypass}$	Crystal reference using the main oscillator (PLL in BYPASS mode)	1	-	8	MHz
$f_{ref\_ext\_bypass}$	External clock reference (PLL in BYPASS mode)	0	-	50	MHz
f_{system_clock}	System clock	0	-	50	MHz

18.2.3 Analog Comparator

Table 18-8. 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 18-9. Analog Comparator Voltage Reference Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
R_{HR}	Resolution high range	-	V_{DD}/32	-	LSB
R_{LR}	Resolution low range	-	V_{DD}/24	-	LSB
A_{HR}	Absolute accuracy high range	-	-	±1/2	LSB
A_{LR}	Absolute accuracy low range	-	-	±1/4	LSB

18.2.4 I^2C

Table 18-10. I^2C Characteristics

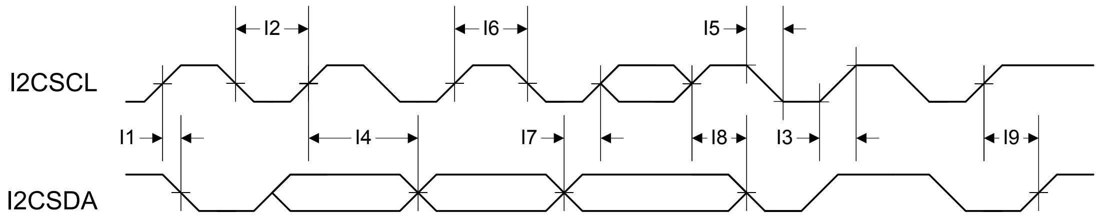
Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
I1[a]	t_{SCH}	Start condition hold time	36	-	-	system clocks
I2[a]	t_{LP}	Clock Low period	36	-	-	system clocks
I3[b]	t_{SRT}	`I2CSCL`/`I2CSDA` rise time (V_{IL} =0.5 V to V_{IH} =2.4 V)	-	-	(see note b)	ns
I4[a]	t_{DH}	Data hold time	2	-	-	system clocks
I5[c]	t_{SFT}	`I2CSCL`/`I2CSDA` fall time (V_{IH} =2.4 V to V_{IL} =0.5 V)	-	9	10	ns
I6[a]	t_{HT}	Clock High time	24	-	-	system clocks
I7[a]	t_{DS}	Data setup time	18	-	-	system clocks
I8[a]	t_{SCSR}	Start condition setup time (for repeated start condition only)	36	-	-	system clocks
I9[a]	t_{SCS}	Stop condition setup time	24	-	-	system clocks

a. Values depend on the value programmed into the `TPR` bit in the **I^2C Master Timer Period (I2CMTPR)** register; a `TPR` programmed for the maximum `I2CSCL` frequency (TPR=0x2) results in a minimum output timing as shown in the table above. The I^2C interface is designed to scale the actual data transition time to move it to the middle of the `I2CSCL` Low period. The actual position is affected by the value programmed into the `TPR`; however, the numbers given in the above values are minimum values.

b. Because `I2CSCL` and `I2CSDA` are open-drain-type outputs, which the controller can only actively drive Low, the time `I2CSCL` or `I2CSDA` takes to reach a high level depends on external signal capacitance and pull-up resistor values.

c. Specified at a nominal 50 pF load.

Figure 18-2. I²C Timing



18.2.5 Synchronous Serial Interface (SSI)

Table 18-11. SSI Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
S1	t_{clk_per}	`SSIClk` cycle time	2	-	65024	system clocks
S2	t_{clk_high}	`SSIClk` high time	-	1/2	-	t clk_per
S3	t_{clk_low}	`SSIClk` low time	-	1/2	-	t clk_per
S4	t_{clkrf}	`SSIClk` rise/fall time	-	7.4	26	ns
S5	t_{DMd}	Data from master valid delay time	0	-	20	ns
S6	t_{DMs}	Data from master setup time	20	-	-	ns
S7	t_{DMh}	Data from master hold time	40	-	-	ns
S8	t_{DSs}	Data from slave setup time	20	-	-	ns
S9	t_{DSh}	Data from slave hold time	40	-	-	ns

Figure 18-3. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement

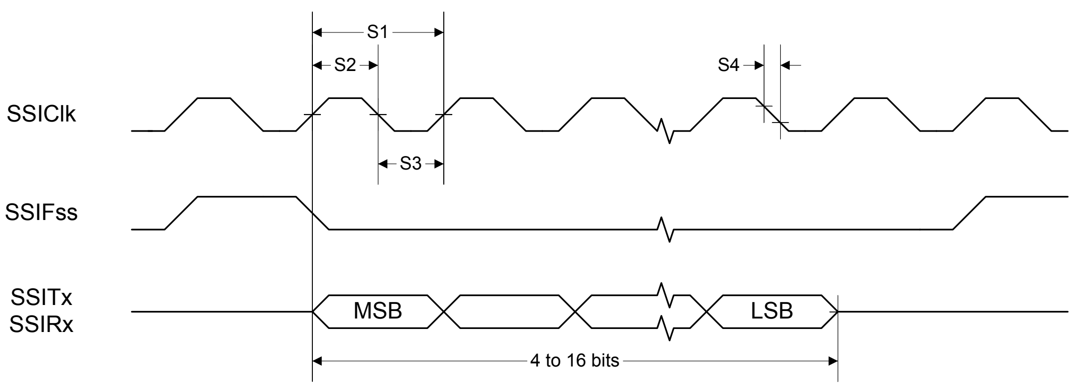


Figure 18-4. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer

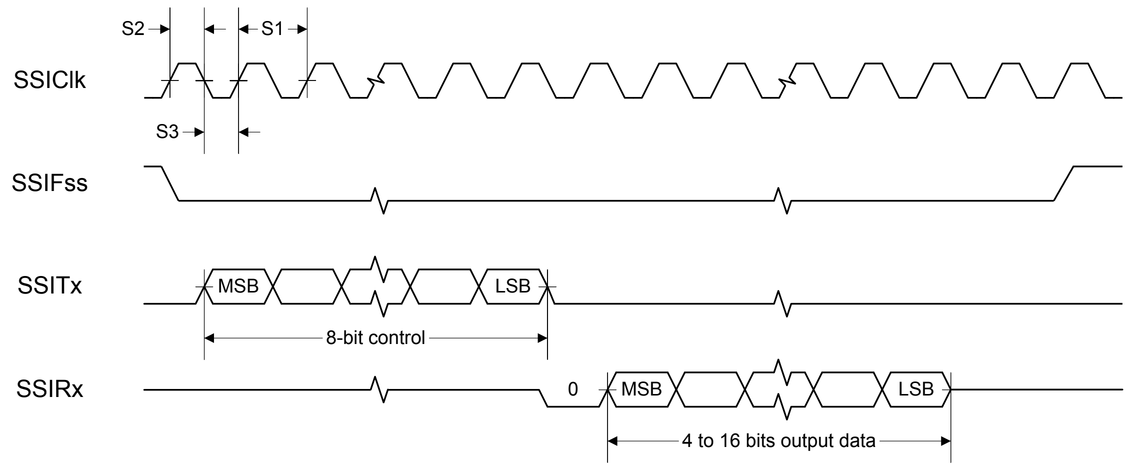
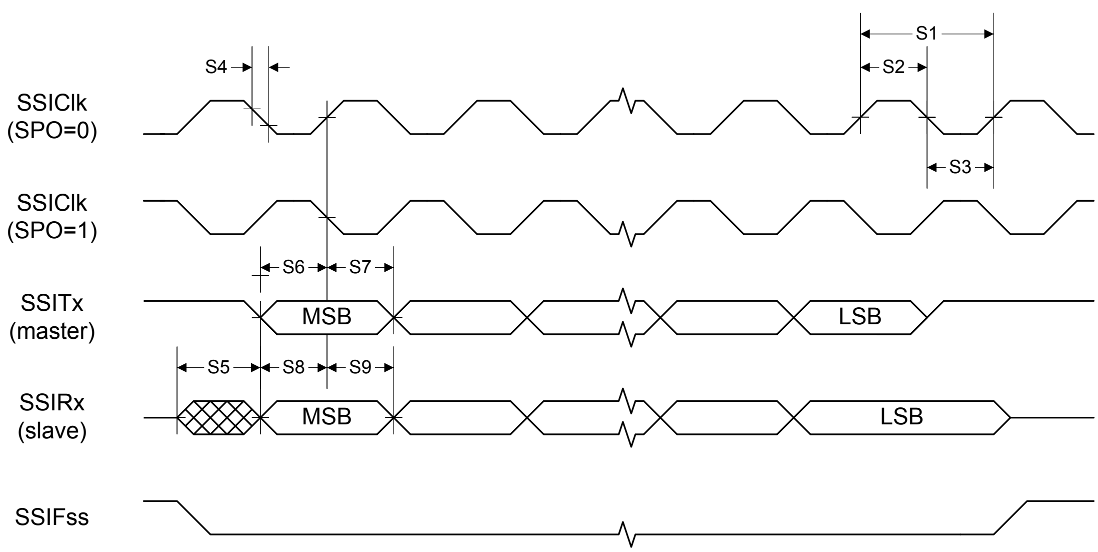


Figure 18-5. SSI Timing for SPI Frame Format (FRF=00), with SPH=1



18.2.6 JTAG and Boundary Scan

Table 18-12. JTAG Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J1	f_{TCK}	TCK operational clock frequency	0	-	10	MHz
J2	t_{TCK}	TCK operational clock period	100	-	-	ns
J3	t_{TCK_LOW}	TCK clock Low time	-	t_{TCK}	-	ns

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
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
J14	t_{TRST}	$\overline{TRST}$ assertion time	100	-	-	ns
J15	t_{TRST_SU}	$\overline{TRST}$ setup time to TCK rise	10	-	-	ns

Figure 18-6. JTAG Test Clock Input Timing

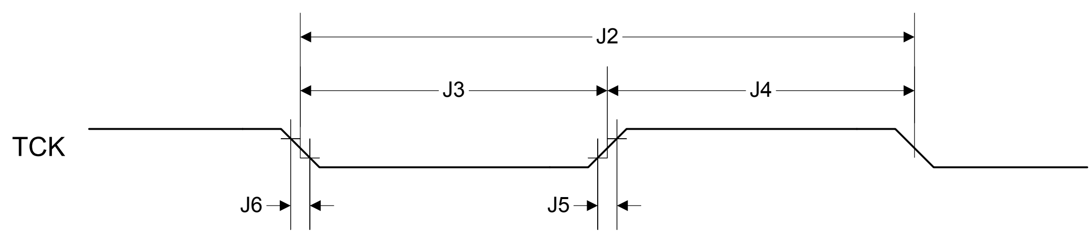


Figure 18-7. JTAG Test Access Port (TAP) Timing

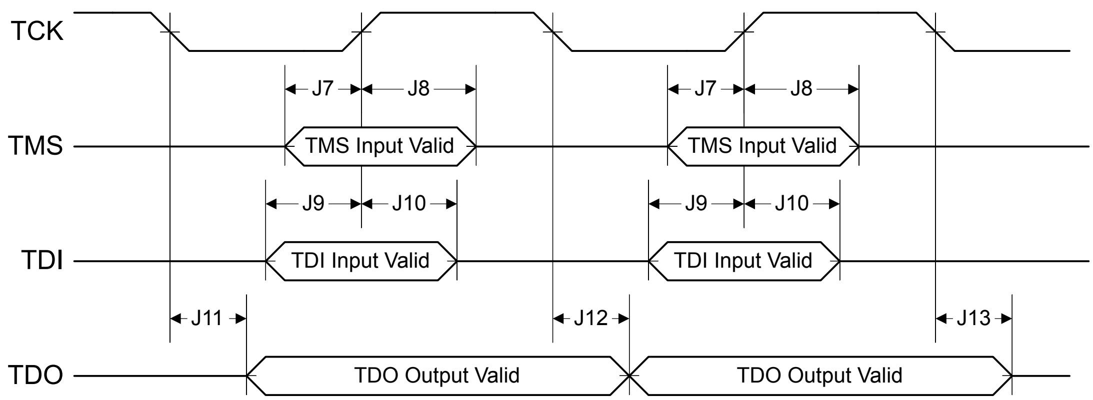
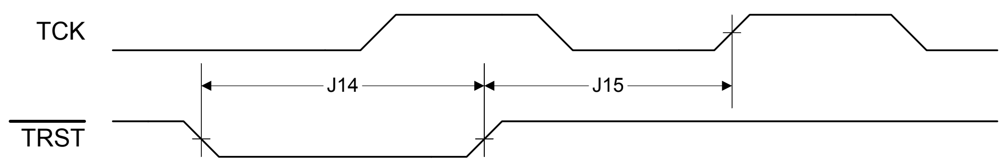


Figure 18-8. JTAG TRST Timing



18.2.7 General-Purpose I/O

Note: All GPIOs are 5 V-tolerant.

Table 18-13. GPIO Characteristics

Parameter	Parameter Name	Condition	Min	Nom	Max	Unit
t_{GPIOR}	GPIO Rise Time (from 20% to 80% of V_{DD})	2-mA drive	-	17	26	ns
		4-mA drive		9	13	ns
		8-mA drive		6	9	ns
		8-mA drive with slew rate control		10	12	ns
t_{GPIOF}	GPIO Fall Time (from 80% to 20% of V_{DD})	2-mA drive	-	17	25	ns
		4-mA drive		8	12	ns
		8-mA drive		6	10	ns
		8-mA drive with slew rate control		11	13	ns

18.2.8 Reset

Table 18-14. Reset Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
R1	V_{TH}	Reset threshold	-	2.0	-	V

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
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	15	-	30	ms
R6	T_{IRBOR}	Internal reset timeout after BOR[a]	2.5	-	20	µs
R7	T_{IRHWR}	Internal reset timeout after hardware reset ($\overline{RST}$ pin)	15	-	30	ms
R8	T_{IRSWR}	Internal reset timeout after software-initiated system reset [a]	2.5	-	20	µs
R9	T_{IRWDR}	Internal reset timeout after watchdog reset[a]	2.5	-	20	µs
R10	T_{IRLDOR}	Internal reset timeout after LDO reset[a]	2.5	-	20	µs
R11	$T_{VDDRISE}$	Supply voltage (V_{DD}) rise time (0 V-3.3 V)	-	-	100	ms

a. 20 * t_{MOSC_per}

Figure 18-9. External Reset Timing ($\overline{RST}$)

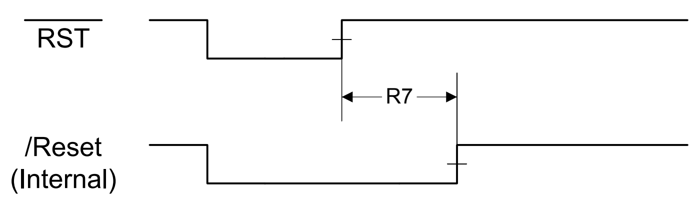


Figure 18-10. Power-On Reset Timing

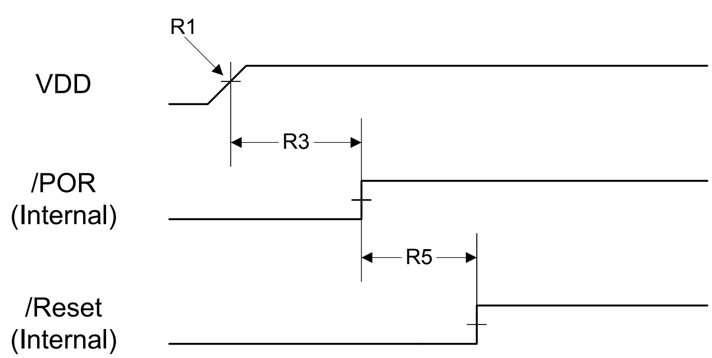


Figure 18-11. Brown-Out Reset Timing

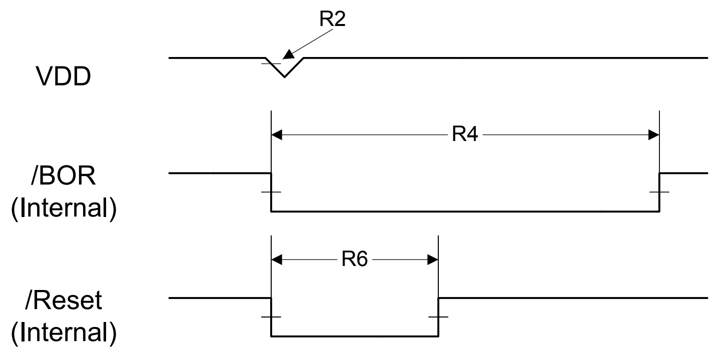


Figure 18-12. Software Reset Timing

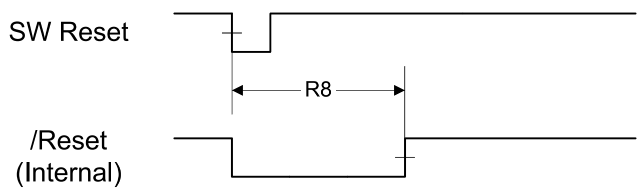


Figure 18-13. Watchdog Reset Timing

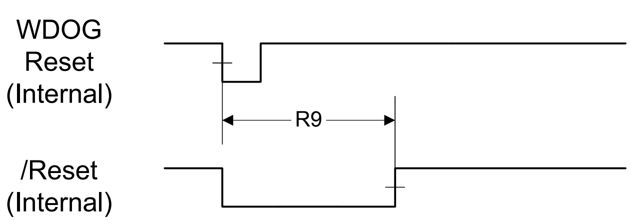
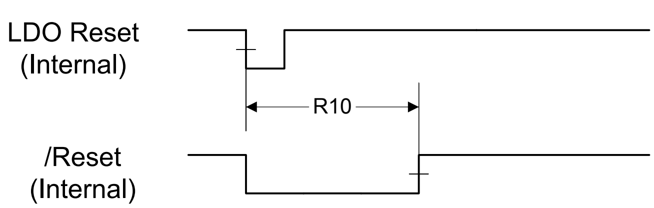
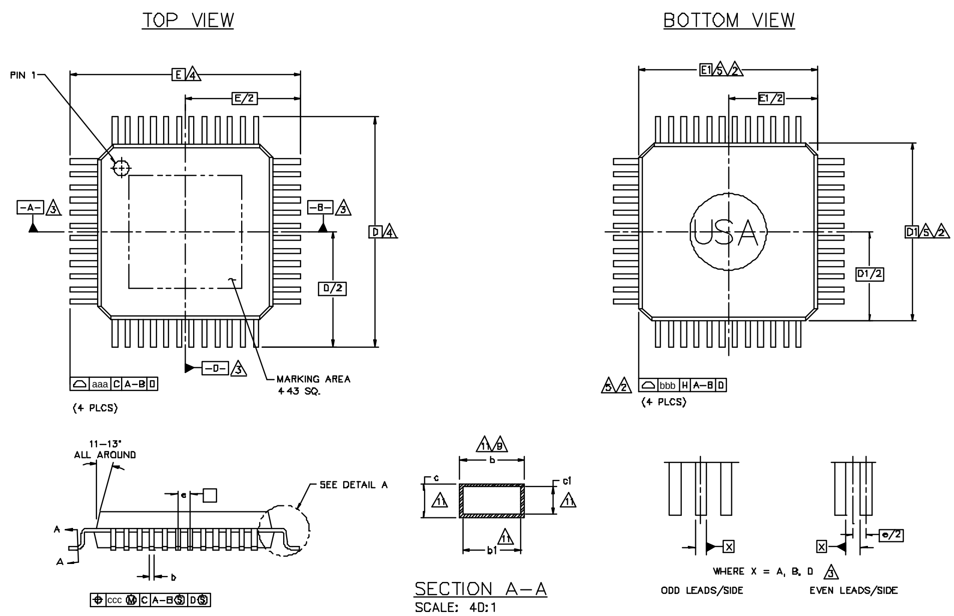


Figure 18-14. LDO Reset Timing



19 Package Information

Figure 19-1. 48-Pin LQFP Package



Note: The following notes apply to the package drawing.

1. All dimensions are in mm. All dimensioning and tolerancing conform to ANSI Y14.5M-1982.
2. The top package body size may be smaller than the bottom package body size by as much as 0.20.
3. Datums A-B and -D- to be determined at datum plane -H-.
4. To be determined at seating plane -C-.
5. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 per side. D1 and E1 are maximum plastic body size dimensions including mold mismatch.
6. Surface finish of the package is #24-27 Charmille (1.6-2.3µmR0) Pin 1 and ejector pin may be less than 0.1µmR0.

7. Dambar removal protrusion does not exceed 0.08. Intrusion does not exceed 0.03.

8. Burr does not exceed 0.08 in any direction.

9. Dimension b does not include Dambar protrusion. Allowable Dambar protrusion shall not cause the lead width to exceed the maximum b dimension by more than 0.08. Dambar cannot be located on the lower radius or the foot. Minimum space between protrusion and adjacent lead is 0.07 for 0.40 and 0.50 pitch package.

10. Corner radius of plastic body does not exceed 0.20.

11. These dimensions apply to the flat section of the lead between 0.10 and 0.25 from the lead tip.

12. A1 is defined as the distance from the seating plane to the lowest point of the package body.

13. Finish of leads is tin plated.

14. All specifications and dimensions are subjected to IPAC'S manufacturing process flow and materials.

15. M5-026A. Where discrepancies between the JEDEC and IPAC documents exist, this drawing will take the precedence.

Symbol	Package Type			Note
	48LD LQFP			
	MIN	NOM	MAX	
A	===	===	1.60	
A_1	0.05	===	0.15	
A_2	1.35	1.40	1.45	
D	9.00 BSC			
D_1	7.00 BSC			
E	9.00 BSC			
E_1	7.00 BSC			
L	0.45	0.80	0.75	
e	0.50 BSC			
b	0.17	0.22	0.27	
b1	0.17	0.20	0.23	
c	0.09	===	0.20	
c1	0.09	===	0.16	
	Tolerances of form and position			
aaa	0.20			
bbb	0.20			
ccc	0.08			
ddd	0.08			

A Serial Flash Loader

A.1 Serial Flash Loader

The Stellaris® serial flash loader is a preprogrammed flash-resident utility used to download code to the flash memory of a device without the use of a debug interface. The serial flash loader uses a simple packet interface to provide synchronous communication with the device. The flash loader runs off the crystal and does not enable the PLL, so its speed is determined by the crystal used. The two serial interfaces that can be used are the UART0 and SSI0 interfaces. For simplicity, both the data format and communication protocol are identical for both serial interfaces.

A.2 Interfaces

Once communication with the flash loader is established via one of the serial interfaces, that interface is used until the flash loader is reset or new code takes over. For example, once you start communicating using the SSI port, communications with the flash loader via the UART are disabled until the device is reset.

A.2.1 UART

The Universal Asynchronous Receivers/Transmitters (UART) communication uses a fixed serial format of 8 bits of data, no parity, and 1 stop bit. The baud rate used for communication is automatically detected by the flash loader and can be any valid baud rate supported by the host and the device. The auto detection sequence requires that the baud rate should be no more than 1/32 the crystal frequency of the board that is running the serial flash loader. This is actually the same as the hardware limitation for the maximum baud rate for any UART on a Stellaris® device which is calculated as follows:

```
Max Baud Rate = System Clock Frequency / 16
```

In order to determine the baud rate, the serial flash loader needs to determine the relationship between its own crystal frequency and the baud rate. This is enough information for the flash loader to configure its UART to the same baud rate as the host. This automatic baud-rate detection allows the host to use any valid baud rate that it wants to communicate with the device.

The method used to perform this automatic synchronization relies on the host sending the flash loader two bytes that are both 0x55. This generates a series of pulses to the flash loader that it can use to calculate the ratios needed to program the UART to match the host's baud rate. After the host sends the pattern, it attempts to read back one byte of data from the UART. The flash loader returns the value of 0xCC to indicate successful detection of the baud rate. If this byte is not received after at least twice the time required to transfer the two bytes, the host can resend another pattern of 0x55, 0x55, and wait for the 0xCC byte again until the flash loader acknowledges that it has received a synchronization pattern correctly. For example, the time to wait for data back from the flash loader should be calculated as at least 2*(20(bits/sync)/baud rate (bits/sec)). For a baud rate of 115200, this time is 2*(20/115200) or 0.35 ms.

A.2.2 SSI

The Synchronous Serial Interface (SSI) port also uses a fixed serial format for communications, with the framing defined as Motorola format with SPH set to 1 and SPO set to 1. See "Frame Formats" on page 256 in the SSI chapter for more information on formats for this transfer protocol. Like the UART, this interface has hardware requirements that limit the maximum speed that the SSI clock can run. This allows the SSI clock to be at most 1/12 the crystal frequency of the board running the flash loader. Since the host device is the master, the SSI on the flash loader device does not need to determine the clock as it is provided directly by the host.

A.3 Packet Handling

All communications, with the exception of the UART auto-baud, are done via defined packets that are acknowledged (ACK) or not acknowledged (NAK) by the devices. The packets use the same format for receiving and sending packets, including the method used to acknowledge successful or unsuccessful reception of a packet.

A.3.1 Packet Format

All packets sent and received from the device use the following byte-packed format.

```
struct
{
 unsigned char ucSize;
 unsigned char ucCheckSum;
 unsigned char Data[];
};
```

ucSize	The first byte received holds the total size of the transfer including the size and checksum bytes.
ucChecksum	This holds a simple checksum of the bytes in the data buffer only. The algorithm is Data[0]+Data[1]+…+ Data[ucSize-3].
Data	This is the raw data intended for the device, which is formatted in some form of command interface. There should be ucSize–2 bytes of data provided in this buffer to or from the device.

A.3.2 Sending Packets

The actual bytes of the packet can be sent individually or all at once; the only limitation is that commands that cause flash memory access should limit the download sizes to prevent losing bytes during flash programming. This limitation is discussed further in the section that describes the serial flash loader command, COMMAND_SEND_DATA (see “COMMAND_SEND_DATA (0x24)” on page 363).

Once the packet has been formatted correctly by the host, it should be sent out over the UART or SSI interface. Then the host should poll the UART or SSI interface for the first non-zero data returned from the device. The first non-zero byte will either be an ACK (0xCC) or a NAK (0x33) byte from the device indicating the packet was received successfully (ACK) or unsuccessfully (NAK). This does not indicate that the actual contents of the command issued in the data portion of the packet were valid, just that the packet was received correctly.

A.3.3 Receiving Packets

The flash loader sends a packet of data in the same format that it receives a packet. The flash loader may transfer leading zero data before the first actual byte of data is sent out. The first non-zero byte is the size of the packet followed by a checksum byte, and finally followed by the data itself. There is no break in the data after the first non-zero byte is sent from the flash loader. Once the device communicating with the flash loader receives all the bytes, it must either ACK or NAK the packet to indicate that the transmission was successful. The appropriate response after sending a NAK to the flash loader is to resend the command that failed and request the data again. If needed, the host may send leading zeros before sending down the ACK/NAK signal to the flash loader, as the flash loader only accepts the first non-zero data as a valid response. This zero padding is needed by the SSI interface in order to receive data to or from the flash loader.

A.4 Commands

The next section defines the list of commands that can be sent to the flash loader. The first byte of the data should always be one of the defined commands, followed by data or parameters as determined by the command that is sent.

A.4.1 COMMAND_PING (0X20)

This command simply accepts the command and sets the global status to success. The format of the packet is as follows:

```
Byte[0] = 0x03;
Byte[1] = checksum(Byte[2]);
Byte[2] = COMMAND_PING;
```

The ping command has 3 bytes and the value for `COMMAND_PING` is 0x20 and the checksum of one byte is that same byte, making Byte[1] also 0x20. Since the ping command has no real return status, the receipt of an ACK can be interpreted as a successful ping to the flash loader.

A.4.2 COMMAND_GET_STATUS (0x23)

This command returns the status of the last command that was issued. Typically, this command should be sent after every command to ensure that the previous command was successful or to properly respond to a failure. The command requires one byte in the data of the packet and should be followed by reading a packet with one byte of data that contains a status code. The last step is to ACK or NAK the received data so the flash loader knows that the data has been read.

```
Byte[0] = 0x03
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_GET_STATUS
```

A.4.3 COMMAND_DOWNLOAD (0x21)

This command is sent to the flash loader to indicate where to store data and how many bytes will be sent by the `COMMAND_SEND_DATA` commands that follow. The command consists of two 32-bit values that are both transferred MSB first. The first 32-bit value is the address to start programming data into, while the second is the 32-bit size of the data that will be sent. This command also triggers an erase of the full area to be programmed so this command takes longer than other commands. This results in a longer time to receive the ACK/NAK back from the board. This command should be followed by a `COMMAND_GET_STATUS` to ensure that the Program Address and Program size are valid for the device running the flash loader.

The format of the packet to send this command is a follows:

```
Byte[0] = 11
Byte[1] = checksum(Bytes[2:10])
Byte[2] = COMMAND_DOWNLOAD
Byte[3] = Program Address [31:24]
Byte[4] = Program Address [23:16]
Byte[5] = Program Address [15:8]
Byte[6] = Program Address [7:0]
Byte[7] = Program Size [31:24]
```

```
Byte[8] = Program Size [23:16]
Byte[9] = Program Size [15:8]
Byte[10] = Program Size [7:0]
```

A.4.4 COMMAND_SEND_DATA (0x24)

This command should only follow a `COMMAND_DOWNLOAD` command or another `COMMAND_SEND_DATA` command if more data is needed. Consecutive send data commands automatically increment address and continue programming from the previous location. The caller should limit transfers of data to a maximum 8 bytes of packet data to allow the flash to program successfully and not overflow input buffers of the serial interfaces. The command terminates programming once the number of bytes indicated by the `COMMAND_DOWNLOAD` command has been received. Each time this function is called it should be followed by a `COMMAND_GET_STATUS` to ensure that the data was successfully programmed into the flash. If the flash loader sends a NAK to this command, the flash loader does not increment the current address to allow retransmission of the previous data.

```
Byte[0] = 11
Byte[1] = checksum(Bytes[2:10])
Byte[2] = COMMAND_SEND_DATA
Byte[3] = Data[0]
Byte[4] = Data[1]
Byte[5] = Data[2]
Byte[6] = Data[3]
Byte[7] = Data[4]
Byte[8] = Data[5]
Byte[9] = Data[6]
Byte[10] = Data[7]
```

A.4.5 COMMAND_RUN (0x22)

This command is used to tell the flash loader to execute from the address passed as the parameter in this command. This command consists of a single 32-bit value that is interpreted as the address to execute. The 32-bit value is transmitted MSB first and the flash loader responds with an ACK signal back to the host device before actually executing the code at the given address. This allows the host to know that the command was received successfully and the code is now running.

```
Byte[0] = 7
Byte[1] = checksum(Bytes[2:6])
Byte[2] = COMMAND_RUN
Byte[3] = Execute Address[31:24]
Byte[4] = Execute Address[23:16]
Byte[5] = Execute Address[15:8]
Byte[6] = Execute Address[7:0]
```

A.4.6 COMMAND_RESET (0x25)

This command is used to tell the flash loader device to reset. This is useful when downloading a new image that overwrote the flash loader and wants to start from a full reset. Unlike the `COMMAND_RUN` command, this allows the initial stack pointer to be read by the hardware and set up for the new code. It can also be used to reset the flash loader if a critical error occurs and the host device wants to restart communication with the flash loader.

```
Byte[0] = 3
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_RESET
```

The flash loader responds with an ACK signal back to the host device before actually executing the software reset to the device running the flash loader. This allows the host to know that the command was received successfully and the part will be reset.

B Register Quick Reference

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
System Control Base 0x400F.E000															
DID0, type RO, offset 0x000, reset -															
	VER														
MAJOR								MINOR							
PBORCTL, type R/W, offset 0x030, reset 0x0000.7FFD															
BORTIM														BORIOR	BORWT
LDOPCTL, type R/W, offset 0x034, reset 0x0000.0000															
										VADJ					
RIS, type RO, offset 0x050, reset 0x0000.0000															
									PLLLRIS	CLRIS	IOFRIS	MOFRIS	LDORIS	BORRIS	PLLFRIS
IMC, type R/W, offset 0x054, reset 0x0000.0000															
									PLLLIM	CLIM	IOFIM	MOFIM	LDOIM	BORIM	PLLFIM
MISC, type R/W1C, offset 0x058, reset 0x0000.0000															
									PLLLMIS	CLMIS	IOFMIS	MOFMIS	LDOMIS	BORMIS	
RESC, type R/W, offset 0x05C, reset -															
										LDO	SW	WDT	BOR	POR	EXT
RCC, type R/W, offset 0x060, reset 0x07A0.3AD1															
				ACG	SYSDIV				USESYSDIV						
		PWRDN	OEN	BYPASS	PLLVER	XTAL				OSCSRC		IOSCVER	MOSCVER	IOSCDIS	MOSCDIS
PLLCFG, type RO, offset 0x064, reset -															
OD		F									R				
DSLPCLKCFG, type R/W, offset 0x144, reset 0x0780.0000															
															IOSC
CLKVCLR, type R/W, offset 0x150, reset 0x0000.0000															
															VERCLR
LDOARST, type R/W, offset 0x160, reset 0x0000.0000															
															LDOARST
DID1, type RO, offset 0x004, reset -															
VER				FAM				PARTNO							
								TEMP			PKG		ROHS	QUAL	
DC0, type RO, offset 0x008, reset 0x001F.000F															
SRAMSZ															
FLASHSZ															
DC1, type RO, offset 0x010, reset 0x0000.309F															
MINSYSDIV								MPU			PLL	WDT	SWO	SWD	JTAG
DC2, type RO, offset 0x014, reset 0x0707.1013															
					COMP2	COMP1	COMP0						TIMER2	TIMER1	TIMER0
			I2C0								SSI0			UART1	UART0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DC3, type RO, offset 0x018, reset 0x3F00.7FC0															
		CCP5	CCP4	CCP3	CCP2	CCP1	CCP0								
	C2O	C2PLUS	C2MINUS	C1O	C1PLUS	C1MINUS	C0O	C0PLUS	C0MINUS						
DC4, type RO, offset 0x01C, reset 0x0000.001F															
											GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
RCGC0, type R/W, offset 0x100, reset 0x00000040															
												WDT			
SCGC0, type R/W, offset 0x110, reset 0x00000040															
												WDT			
DCGC0, type R/W, offset 0x120, reset 0x00000040															
												WDT			
RCGC1, type R/W, offset 0x104, reset 0x00000000															
					COMP2	COMP1	COMP0						TIMER2	TIMER1	TIMER0
			I2C0								SSI0			UART1	UART0
SCGC1, type R/W, offset 0x114, reset 0x00000000															
					COMP2	COMP1	COMP0						TIMER2	TIMER1	TIMER0
			I2C0								SSI0			UART1	UART0
DCGC1, type R/W, offset 0x124, reset 0x00000000															
					COMP2	COMP1	COMP0						TIMER2	TIMER1	TIMER0
			I2C0								SSI0			UART1	UART0
RCGC2, type R/W, offset 0x108, reset 0x00000000															
											GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
SCGC2, type R/W, offset 0x118, reset 0x00000000															
											GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
DCGC2, type R/W, offset 0x128, reset 0x00000000															
											GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
SRCR0, type R/W, offset 0x040, reset 0x00000000															
												WDT			
SRCR1, type R/W, offset 0x044, reset 0x00000000															
					COMP2	COMP1	COMP0						TIMER2	TIMER1	TIMER0
			I2C0								SSI0			UART1	UART0
SRCR2, type R/W, offset 0x048, reset 0x00000000															
											GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Internal Memory															
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															

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
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
Internal Memory **System Control Offset** Base 0x400F.E000															
USECRL, type R/W, offset 0x140, reset 0x31															
								USEC							
FMPRE, type R/W, offset 0x130, reset 0x8000.FFFF															
READ_ENABLE															
READ_ENABLE															
FMPPE, type R/W, offset 0x134, reset 0x0000.FFFF															
PROG_ENABLE															
PROG_ENABLE															
General-Purpose Input/Outputs (GPIOs) GPIO Port A base: 0x4000.4000 GPIO Port B base: 0x4000.5000 GPIO Port C base: 0x4000.6000 GPIO Port D base: 0x4000.7000 GPIO Port E base: 0x4002.4000															
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							
GPIOMIS, type RO, offset 0x418, reset 0x0000.0000															
								MIS							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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 0x0000.00FF															
								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 0x0000.00FF															
								DEN							
GPIOPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000															
								PID4							
GPIOPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000															
								PID5							
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							

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
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															
												TAAMS	TACMR	TAMR	
GPTMTBMR, type R/W, offset 0x008, reset 0x0000.0000															
												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															
					CBEIM	CBMIM	TBTOIM					RTCIM	CAEIM	CAMIM	TATOIM
GPTMRIS, type RO, offset 0x01C, reset 0x0000.0000															
					CBERIS	CBMRIS	TBTORIS					RTCRIS	CAERIS	CAMRIS	TATORIS
GPTMMIS, type RO, offset 0x020, reset 0x0000.0000															
					CBEMIS	CBMMIS	TBTOMIS					RTCMIS	CAEMIS	CAMMIS	TATOMIS
GPTMICR, type W1C, offset 0x024, reset 0x0000.0000															
					CBECINT	CBMCINT	TBTOCINT					RTCCINT	CAECINT	CAMCINT	TATOCINT
GPTMTAILR, type R/W, offset 0x028, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)															
TAILRH															
TAILRL															
GPTMTBILR, type R/W, offset 0x02C, reset 0x0000.FFFF															
TBILRL															
GPTMTAMATCHR, type R/W, offset 0x030, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)															
TAMRH															
TAMRL															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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							
GPTMTAPMR, type R/W, offset 0x040, reset 0x0000.0000															
								TAPSMR							
GPTMTBPMR, type R/W, offset 0x044, reset 0x0000.0000															
								TBPSMR							
GPTMTAR, type RO, offset 0x048, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)															
TARH															
TARL															
GPTMTBR, type RO, offset 0x04C, reset 0x0000.FFFF															
TBRL															
Watchdog Timer Base 0x4000.0000															
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															
														RESEN	INTEN
WDTICR, type WO, offset 0x00C, reset -															
WDTIntClr															
WDTIntClr															
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							

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
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 0x0000.0005															
								CID2							
WDTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1															
								CID3							
Universal Asynchronous Receivers/Transmitters (UARTs) UART0 base: 0x4000.C000 UART1 base: 0x4000.D000															
UARTDR, type R/W, offset 0x000, reset 0x0000.0000															
				OE	BE	PE	FE	DATA							
UARTRSR/UARTECR, type RO, offset 0x004, reset 0x0000.0000															
												OE	BE	PE	FE
UARTRSR/UARTECR, type WO, offset 0x004, reset 0x0000.0000															
								DATA							
UARTFR, type RO, offset 0x018, reset 0x0000.0090															
								TXFE	RXFF	TXFF	RXFE	BUSY			
UARTIBRD, type R/W, offset 0x024, reset 0x0000.0000															
DIVINT															
UARTFBRD, type R/W, offset 0x028, reset 0x0000.0000															
										DIVFRAC					

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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							UARTEN
UARTIFLS, type R/W, offset 0x034, reset 0x0000.0012															
										RXIFLSEL			TXIFLSEL		
UARTIM, type R/W, offset 0x038, reset 0x0000.0000															
					OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM				
UARTRIS, type RO, offset 0x03C, reset 0x0000.000F															
					OERIS	BERIS	PERIS	FERIS	RTRIS	TXRIS	RXRIS				
UARTMIS, type RO, offset 0x040, reset 0x0000.0000															
					OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS				
UARTICR, type W1C, offset 0x044, reset 0x0000.0000															
					OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC				
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 0x0000.0011															
								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							

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
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															
SSICR0, type R/W, offset 0x000, reset 0x0000.0000															
SCR								SPH	SPO	FRF		DSS			
SSICR1, type R/W, offset 0x004, reset 0x0000.0000															
												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
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							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSIPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018															
								PID2							
SSIPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001															
								PID3							
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^2C) Interface															
I^2C Master															
I2C Master 0 base: 0x4002.0000															
I2CMSA, type R/W, offset 0x000, reset 0x0000.0000															
								SA							R/S
I2CMCS, type RO, offset 0x004, reset 0x0000.0000															
									BUSBSY	IDLE	ARBLST	DATACK	ADRACK	ERROR	BUSY
I2CMCS, type WO, offset 0x004, reset 0x0000.0000															
												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^2C) Interface															

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
I²C Slave I2C Slave 0 base: 0x4002.0800															
I2CSOAR, type R/W, offset 0x000, reset 0x0000.0000															
									OAR						
I2CSCSR, type RO, offset 0x004, reset 0x0000.0000															
													FBR	TREQ	RREQ
I2CSCSR, type WO, offset 0x004, reset 0x0000.0000															
															DA
I2CSDR, type R/W, offset 0x008, reset 0x0000.0000															
								DATA							
I2CSIMR, type R/W, offset 0x00C, reset 0x0000.0000															
															IM
I2CSRIS, type RO, offset 0x010, reset 0x0000.0000															
															RIS
I2CSMIS, type RO, offset 0x014, reset 0x0000.0000															
															MIS
I2CSICR, type WO, offset 0x018, reset 0x0000.0000															
															IC
Analog Comparators Base 0x4003.C000															
ACMIS, type R/W1C, offset 0x00, reset 0x0000.0000															
													IN2	IN1	IN0
ACRIS, type RO, offset 0x04, reset 0x0000.0000															
													IN2	IN1	IN0
ACINTEN, type R/W, offset 0x08, reset 0x0000.0000															
													IN2	IN1	IN0
ACREFCTL, type R/W, offset 0x10, reset 0x0000.0000															
						EN	RNG					VREF			
ACSTAT0, type RO, offset 0x20, reset 0x0000.0000															
														OVAL	
ACSTAT1, type RO, offset 0x40, reset 0x0000.0000															
														OVAL	
ACSTAT2, type RO, offset 0x60, reset 0x0000.0000															
														OVAL	
ACCTL0, type R/W, offset 0x24, reset 0x0000.0000															
					ASRCP						ISLVAL	ISEN		CINV	

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
ACCTL1, type R/W, offset 0x44, reset 0x0000.0000															
					ASRCP						ISLVAL	ISEN		CINV	
ACCTL2, type R/W, offset 0x64, reset 0x0000.0000															
					ASRCP						ISLVAL	ISEN		CINV	

C Ordering and Contact Information

C.1 Ordering Information

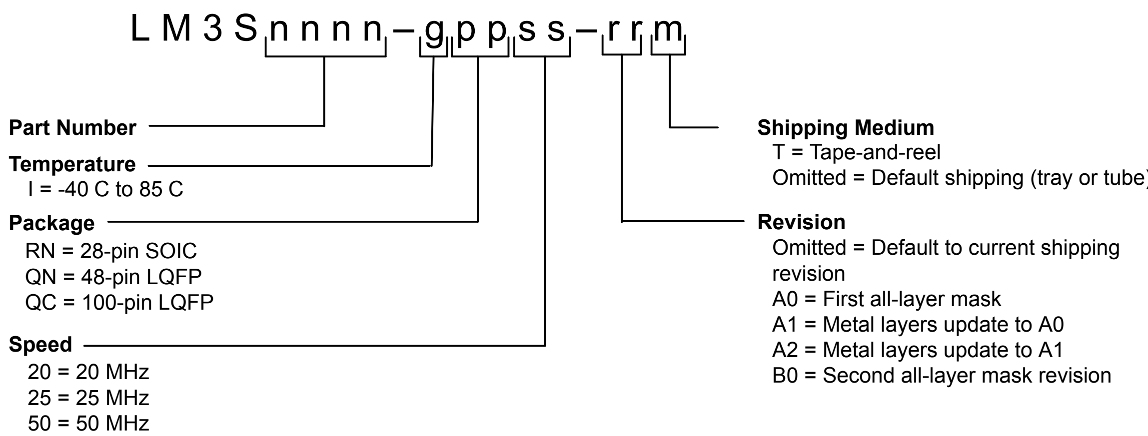


Table C-1. Part Ordering Information

Orderable Part Number	Description
LM3S600-IQN50	Stellaris® LM3S600 Microcontroller
LM3S600-IQN50(T)	Stellaris® LM3S600 Microcontroller

C.2 Kits

The Luminary Micro Stellaris® Family provides the hardware and software tools that engineers need to begin development quickly.

- Reference Design Kits accelerate product development by providing ready-to-run hardware, and comprehensive documentation including hardware design files:

  http://www.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/evaluation_kits/

- Development Kits provide you with all the tools you need to develop and prototype embedded applications right out of the box:

  http://www.luminarymicro.com/products/boards.html

See the Luminary Micro website for the latest tools available or ask your Luminary Micro distributor.

C.3 Company Information

Luminary Micro, Inc. designs, markets, and sells ARM Cortex-M3-based microcontrollers (MCUs). Austin, Texas-based Luminary Micro is the lead partner for the Cortex-M3 processor, delivering the world's first silicon implementation of the Cortex-M3 processor. Luminary Micro's introduction of the

Stellaris® family of products provides 32-bit performance for the same price as current 8- and 16-bit microcontroller designs. With entry-level pricing at $1.00 for an ARM technology-based MCU, Luminary Micro's Stellaris product line allows for standardization that eliminates future architectural upgrades or software tool changes.

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

C.4 Support Information

For support on Luminary Micro products, contact:

support@luminarymicro.com +1-512-279-8800, ext. 3