LUMINARY MICRO®

LM3S5749 Microcontroller

DATA SHEET

DS-LM3S5749-4283

Legal Disclaimers and Trademark Information

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

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

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

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

LUMINARY MICRO

Table of Contents

List of Figures

List of Tables

List of Registers

Revision History

The revision history table notes changes made between the indicated revisions of the LM3S5749 data sheet.

Table 1. Revision History

Date	Revision	Description
June 2008	2972	Started tracking revision history.
October 2008	4149	■ Added note on clearing interrupts to the Interrupts chapter: **Note:** It may take several processor cycles after a write to clear an interrupt source in order for NVIC to see the interrupt source de-assert. This means if the interrupt clear is done as the last action in an interrupt handler, it is possible for the interrupt handler to complete while NVIC sees the interrupt as still asserted, causing the interrupt handler to be re-entered errantly. This can be avoided by either clearing the interrupt source at the beginning of the interrupt handler or by performing a read or write after the write to clear the interrupt source (and flush the write buffer) ■ Added clarification on JTAG reset to the JTAG chapter: In order to reset the JTAG module after the device has been powered on, the TMS input must be held HIGH for five TCK clock cycles, resetting the TAP controller and all associated JTAG chains. ■ The binary value was incorrect in the JTAG 16-bit switch sequence in the JTAG-to-SWD Switching section in the JTAG chapter. Sentence changed to: The 16-bit switch sequence for switching to JTAG mode is defined as b1110011100111100, transmitted LSB first. ■ The FMA value for the **FMPRE3** register was incorrect in the Flash Resident Registers table in the Internal Memory chapter. The correct value is 0x0000.0006. ■ Step 1 of the Initialization and Configuration procedure in the ADC chapter states the wrong register to use to enable the ADC clock. Sentence changed to: 1. Enable the ADC clock by writing a value of 0x0001.0000 to the **RCGC0** register. ■ In the CAN chapter, major improvements were made including a rewrite of the conceptual information and the addition of new figures to clarify how to use the Controller Area Network (CAN) module. ■ In the USB chapter, clarified endpoint terminology and added a new section on DMA Operation. ■ Incorrect Comparator Operating Modes tables were removed from the Analog Comparators chapter. ■ Additional minor data sheet clarifications and corrections were made.
November 2008	4283	■ Revised High-Level Block Diagram. ■ Additional minor data sheet clarifications and corrections were made.

About This Document

This data sheet provides reference information for the LM3S5749 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*
- *Stellaris® Peripheral Driver Library User's Guide*
- *Stellaris® ROM User's Guide*

The following related documents are also referenced:

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

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

Documentation Conventions

This document uses the conventions shown in Table 2 on page 26.

Table 2. Documentation Conventions

Notation	Meaning
General Register Notation	
REGISTER	APB registers are indicated in uppercase bold. For example, **PBORCTL** is the Power-On and Brown-Out Reset Control register. If a register name contains a lowercase n, it represents more than one register. For example, **SRCRn** represents any (or all) of the three Software Reset Control registers: **SRCR0, SRCR1** , and **SRCR2**.
bit	A single bit in a register.
bit field	Two or more consecutive and related bits.
offset 0x*nnn*	A hexadecimal increment to a register's address, relative to that module's base address as specified in “Memory Map” on page 54.

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

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

1 Architectural Overview

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

The Stellaris® family offers efficient performance and extensive integration, favorably positioning the device into cost-conscious applications requiring significant control-processing and connectivity capabilities. The Stellaris® LM3S5000 series combines USB 2.0 Full-Speed On-The-Go/Host/Device combinations with Bosch CAN networking technology.

The LM3S5749 microcontroller is targeted for industrial applications, including remote monitoring, electronic point-of-sale machines, test and measurement equipment, network appliances and switches, factory automation, HVAC and building control, gaming equipment, motion control, medical instrumentation, and fire and security.

For applications requiring extreme conservation of power, the LM3S5749 microcontroller features a battery-backed Hibernation module to efficiently power down the LM3S5749 to a low-power state during extended periods of inactivity. With a power-up/power-down sequencer, a continuous time counter (RTC), a pair of match registers, an APB interface to the system bus, and dedicated non-volatile memory, the Hibernation module positions the LM3S5749 microcontroller perfectly for battery applications.

In addition, the LM3S5749 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 LM3S5749 microcontroller is code-compatible to all members of the extensive Stellaris® family; providing flexibility to fit our customers' precise needs.

Luminary Micro offers a complete solution to get to market quickly, with evaluation and development boards, white papers and application notes, an easy-to-use peripheral driver library, and a strong support, sales, and distributor network. See “Ordering and Contact Information” on page 813 for ordering information for Stellaris® family devices.

1.1 Product Features

The LM3S5749 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
- 39 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

- ARM® Cortex™-M3 Processor Core
  - 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.
  - Deterministic, fast interrupt processing: always 12 cycles, or just 6 cycles with tail-chaining
  - External non-maskable interrupt signal (NMI) available for immediate execution of NMI handler for safety critical applications.
  - 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
    - 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
  - Optimized for single-cycle flash usage
  - Three sleep modes with clock gating for low power
  - Single-cycle multiply instruction and hardware divide

  - Atomic operations
  - ARM Thumb2 mixed 16-/32-bit instruction set
  - 1.25 DMIPS/MHz
- JTAG
  - IEEE 1149.1-1990 compatible Test Access Port (TAP) controller
  - Four-bit Instruction Register (IR) chain for storing JTAG instructions
  - IEEE standard instructions: BYPASS, IDCODE, SAMPLE/PRELOAD, EXTEST and INTEST
  - ARM additional instructions: APACC, DPACC and ABORT
  - Integrated ARM Serial Wire Debug (SWD)
- Hibernation
  - System power control using discrete external regulator
  - Dedicated pin for waking from an external signal
  - Low-battery detection, signaling, and interrupt generation
  - 32-bit real-time counter (RTC)
  - Two 32-bit RTC match registers for timed wake-up and interrupt generation
  - Clock source from a 32.768-kHz external oscillator or a 4.194304-MHz crystal
  - RTC predivider trim for making fine adjustments to the clock rate
  - 64 32-bit words of non-volatile memory
  - Programmable interrupts for RTC match, external wake, and low battery events
- Internal Memory
  - 128 KB single-cycle flash
    - User-managed flash block protection on a 2-KB block basis
    - User-managed flash data programming
    - User-defined and managed flash-protection block
  - 64 KB single-cycle SRAM
  - Pre-programmed ROM
    - Stellaris® family peripheral driver library (DriverLib)
    - Stellaris® boot loader
- DMA Controller

- ARM PrimeCell® 32-channel configurable µDMA controller
- Support for multiple transfer modes
  - Basic, for simple transfer scenarios
  - Ping-pong, for continuous data flow to/from peripherals
  - Scatter-gather, from a programmable list of arbitrary transfers initiated from a single request
- Dedicated channels for supported peripherals
- One channel each for receive and transmit path for bidirectional peripherals
- Dedicated channel for software-initiated transfers
- Independently configured and operated channels
- Per-channel configurable bus arbitration scheme
- Two levels of priority
- Design optimizations for improved bus access performance between µDMA controller and the processor core
  - µDMA controller access is subordinate to core access
  - RAM striping
  - Peripheral bus segmentation
- Data sizes of 8, 16, and 32 bits
- Source and destination address increment size of byte, half-word, word, or no increment
- Maskable device requests
- Optional software initiated requests for any channel
- Interrupt on transfer completion, with a separate interrupt per channel

■ GPIOs

- 0-61 GPIOs, depending on configuration
- 5-V-tolerant input/outputs
- Two means of port access: either high speed (for single-cycle writes), or legacy for backwards-compatibility with existing code
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values

- Bit masking in both read and write operations through address lines
- Can initiate an ADC sample sequence
- Pins configured as digital inputs are Schmitt-triggered.
- Programmable control for GPIO pad configuration
  - Weak pull-up or pull-down resistors
  - 2-mA, 4-mA, and 8-mA pad drive for digital communication; up to four pads can be configured with an 18-mA pad drive for high-current applications
  - Slew rate control for the 8-mA drive
  - Open drain enables
  - Digital input enables

- General-Purpose Timers
  - Four General-Purpose Timer Modules (GPTM), each of which provides two 16-bit timers. Each GPTM can be configured to operate independently:
    - As a single 32-bit timer
    - As one 32-bit Real-Time Clock (RTC) to event capture
    - For Pulse Width Modulation (PWM)
    - To trigger analog-to-digital conversions
  - 32-bit Timer modes
    - Programmable one-shot timer
    - Programmable periodic timer
    - Real-Time Clock when using an external 32.768-KHz clock as the input
    - Software-controlled event stalling (excluding RTC mode)
    - ADC event trigger
  - 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
    - User-enabled stalling when the controller asserts CPU Halt flag during debug
    - ADC event trigger

- 16-bit Input Capture modes
  - Input edge count capture
  - Input edge time capture
- 16-bit PWM mode
  - Simple PWM mode with software-programmable output inversion of the PWM signal

- ARM FiRM-compliant Watchdog Timer
  - 32-bit down counter with a programmable load register
  - Separate watchdog clock with an enable
  - Programmable interrupt generation logic with interrupt masking
  - Lock register protection from runaway software
  - Reset generation logic with an enable/disable
  - User-enabled stalling when the controller asserts the CPU Halt flag during debug
- ADC
  - Eight analog input channels
  - Single-ended and differential-input configurations
  - On-chip internal temperature sensor
  - Sample rate of one million samples/second
  - Flexible, configurable analog-to-digital conversion
  - Four programmable sample conversion sequences from one to eight entries long, with corresponding conversion result FIFOs
  - Flexible trigger control
    - Controller (software)
    - Timers
    - Analog Comparators
    - PWM
    - GPIO
  - Hardware averaging of up to 64 samples for improved accuracy
  - Converter uses an internal 3-V reference
  - Power and ground for the analog circuitry is separate from the digital power and ground

- UART
  - Two fully programmable 16C550-type UARTs with IrDA support
  - Separate 16x8 transmit (TX) and 16x12 receive (RX) FIFOs to reduce CPU interrupt service loading
  - Programmable baud-rate generator allowing speeds up to 3.125 Mbps
  - Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
  - FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
  - Standard asynchronous communication bits for start, stop, and parity
  - False-start bit detection
  - Line-break generation and detection
  - Fully programmable serial interface characteristics
    - 5, 6, 7, or 8 data bits
    - Even, odd, stick, or no-parity bit generation/detection
    - 1 or 2 stop bit generation
  - IrDA serial-IR (SIR) encoder/decoder providing
    - Programmable use of IrDA Serial Infrared (SIR) or UART input/output
    - Support of IrDA SIR encoder/decoder functions for data rates up to 115.2 Kbps half-duplex
    - Support of normal 3/16 and low-power (1.41-2.23 µs) bit durations
    - Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration
  - Dedicated Direct Memory Access (DMA) transmit and receive channels
- Synchronous Serial Interface (SSI)
  - Two SSI modules, each with the following features:
  - Master or slave operation
  - Support for Direct Memory Access (DMA)
  - 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

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

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

- USB
  - Standards-based

- USB 2.0 full-speed (12 Mbps) and low-speed (1.5 Mbps) operation
- USB Host mode
- Integrated PHY
- 4 transfer types: Control, Interrupt, Bulk, and Isochronous
- 8 endpoints
  - 1 dedicated control IN endpoint and 1 dedicated control OUT endpoint
  - 3 configurable IN endpoints and 3 configurable OUT endpoints
- 4 KB dedicated endpoint memory
  - Direct memory access (DMA)
  - One endpoint may be defined for double-buffered 1023-byte isochronous packet size

■ Analog Comparators

- Two independent integrated analog comparators
- Configurable for output to drive an output pin, generate an interrupt, or initiate an ADC sample sequence
- Compare external pin input to external pin input or to internal programmable voltage reference
- 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

■ PWM

- Four PWM generator blocks, each with one 16-bit counter, two PWM comparators, a PWM signal generator, a dead-band generator, and an interrupt/ADC-trigger selector
- Four fault inputs in hardware to promote low-latency shutdown
- One 16-bit counter
  - Runs in Down or Up/Down mode
  - Output frequency controlled by a 16-bit load value
  - Load value updates can be synchronized
  - Produces output signals at zero and load value
- Two PWM comparators
  - Comparator value updates can be synchronized

        - Produces output signals on match
    - PWM generator
        - Output PWM signal is constructed based on actions taken as a result of the counter and PWM comparator output signals
        - Produces two independent PWM signals
    - Dead-band generator
        - Produces two PWM signals with programmable dead-band delays suitable for driving a half-H bridge
        - Can be bypassed, leaving input PWM signals unmodified
    - Flexible output control block with PWM output enable of each PWM signal
        - PWM output enable of each PWM signal
        - Optional output inversion of each PWM signal (polarity control)
        - Optional fault handling for each PWM signal
        - Synchronization of timers in the PWM generator blocks
        - Synchronization of timer/comparator updates across the PWM generator blocks
        - Interrupt status summary of the PWM generator blocks
    - Can initiate an ADC sample sequence
- QEI
    - Position integrator that tracks the encoder position
    - Velocity capture using built-in timer
    - The input frequency of the QEI inputs may be as high as 1/4 of the processor frequency (for example, 12.5 MHz for a 50-MHz system)
    - Interrupt generation on:
        - Index pulse
        - Velocity-timer expiration
        - Direction change
        - Quadrature error detection
- Power
    - On-chip Low Drop-Out (LDO) voltage regulator, with programmable output user-adjustable from 2.25 V to 2.75 V

  - Hibernation module handles the power-up/down 3.3 V sequencing and control for the core digital logic and analog circuits
  - Low-power options on controller: Sleep and Deep-sleep modes
  - Low-power options for peripherals: software controls shutdown of individual peripherals
  - User-enabled LDO unregulated voltage detection and automatic reset
  - 3.3-V supply brown-out detection and reporting via interrupt or reset
- Flexible Reset Sources
  - Power-on reset (POR)
  - Reset pin assertion
  - Brown-out (BOR) detector alerts to system power drops
  - Software reset
  - Watchdog timer reset
  - Internal low drop-out (LDO) regulator output goes unregulated
- Industrial-range 100-pin RoHS-compliant LQFP package

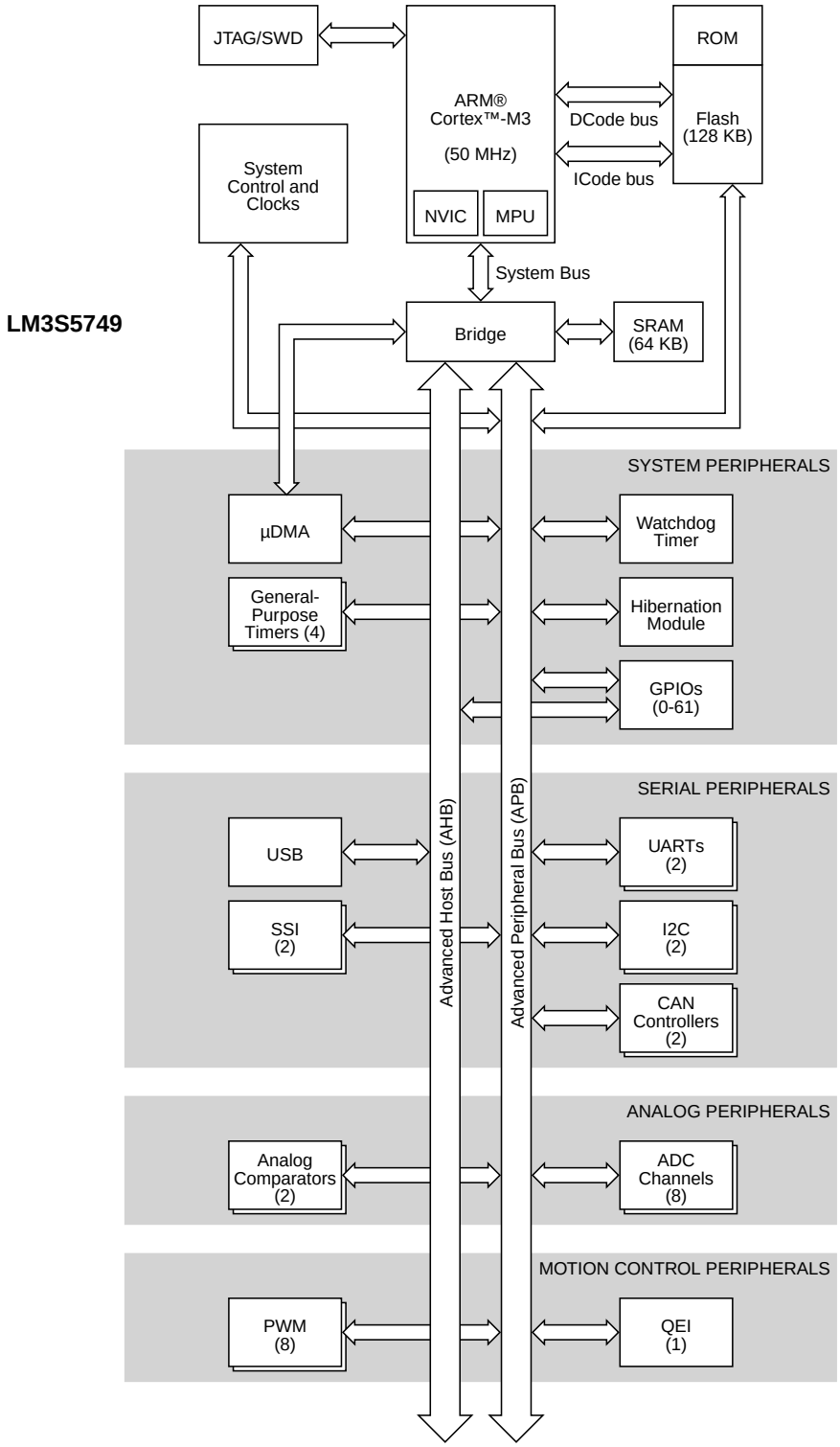
1.2 Target Applications

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

1.3 High-Level Block Diagram

Figure 1-1 on page 40 depicts the features on the Stellaris® LM3S5749 microcontroller.

Figure 1-1. Stellaris® LM3S5749 Microcontroller High-Level Block Diagram



1.4 Functional Overview

The following sections provide an overview of the features of the LM3S5749 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 813.

1.4.1 ARM Cortex™-M3

1.4.1.1 Processor Core (see page 48)

All members of the Stellaris® product family, including the LM3S5749 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 48 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) (see page 51)

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) (see page 57)

The LM3S5749 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 39 interrupts.

“Interrupts” on page 57 provides an overview of the NVIC controller and the interrupt map. Exceptions and interrupts are detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

1.4.1.4 Direct Memory Access (see page 194)

The LM3S5749 microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (µDMA). The µDMA controller provides a way to offload data transfer tasks from the

Cortex-M3 processor, allowing for more efficient use of the processor and the expanded available bus bandwidth. The µDMA controller can perform transfers between memory and peripherals. It has dedicated channels for each supported peripheral and can be programmed to automatically perform transfers between peripherals and memory as the peripheral is ready to transfer more data. The µDMA controller also supports sophisticated transfer modes such as ping-pong and scatter-gather, which allows the processor to set up a list of transfer tasks for the controller.

1.4.2 Motor Control Peripherals

To enhance motor control, the LM3S5749 controller features Pulse Width Modulation (PWM) outputs and the Quadrature Encoder Interface (QEI).

1.4.2.1 PWM

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

On the LM3S5749, PWM motion control functionality can be achieved through:

- Dedicated, flexible motion control hardware using the PWM pins
- The motion control features of the general-purpose timers using the CCP pins

PWM Pins (see page 660)

The LM3S5749 PWM module consists of four PWM generator blocks and a control block. Each PWM generator block contains one timer (16-bit down or up/down counter), two comparators, a PWM signal generator, a dead-band generator, and an interrupt/ADC-trigger selector. The control block determines the polarity of the PWM signals, and which signals are passed through to the pins.

Each PWM generator block produces two PWM signals that can either be independent signals or a single pair of complementary signals with dead-band delays inserted. The output of the PWM generation blocks are managed by the output control block before being passed to the device pins.

CCP Pins (see page 309)

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.

Fault Pins ((see page 666))

The LM3S5749 PWM module includes four fault-condition handling inputs to quickly provide low-latency shutdown and prevent damage to the motor being controlled.

1.4.2.2 QEI (see page 715)

A quadrature encoder, also known as a 2-channel incremental encoder, converts linear displacement into a pulse signal. By monitoring both the number of pulses and the relative phase of the two signals, you can track the position, direction of rotation, and speed. In addition, a third channel, or index signal, can be used to reset the position counter.

The Stellaris quadrature encoder with index (QEI) module interprets the code produced by a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, it can capture a running estimate of the velocity of the encoder wheel.

1.4.3 Analog Peripherals

To handle analog signals, the LM3S5749 microcontroller offers an Analog-to-Digital Converter (ADC).

For support of analog signals, the LM3S5749 microcontroller offers two analog comparators.

1.4.3.1 ADC (see page 361)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number.

The LM3S5749 ADC module features 10-bit conversion resolution and supports eight input channels, plus an internal temperature sensor. Four buffered sample sequences allow rapid sampling of up to eight analog input sources without controller intervention. Each sample sequence provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequence priority.

1.4.3.2 Analog Comparators (see page 648)

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

The LM3S5749 microcontroller provides two independent integrated analog comparators that can be configured to drive an output or generate an interrupt or ADC event.

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

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

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

1.4.4 Serial Communications Peripherals

The LM3S5749 controller supports both asynchronous and synchronous serial communications with:

- Two fully programmable 16C550-type UARTs
- Two SSI modules
- Two I^2C modules
- One USB 2.0 full-speed controller
- Two CAN units

1.4.4.1 UART (see page 396)

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 LM3S5749 controller includes two fully programmable 16C550-type UARTs that support data transfer speeds up to 3.125 Mbps. (Although similar in functionality to a 16C550 UART, it is not register-compatible.) In addition, each UART is capable of supporting IrDA.

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 439)

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

The LM3S5749 controller includes two SSI modules that provide 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.

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

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

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

1.4.4.3 I^2C (see page 479)

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 LM3S5749 controller includes two I^2C modules that provide 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. Each 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.

1.4.4.4 USB (see page 561)

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

The LM3S5749 controller supports the USB 2.0 full-speed configuration with Device or USB Host mode. The specified throughput for a USB 2.0 full-speed controller is 12 Mbps.

1.4.4.5 Controller Area Network (see page 515)

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

A transmitter sends a message to all CAN nodes (broadcasting). Each node decides on the basis of the identifier received whether it should process the message. The identifier also determines the priority that the message enjoys in competition for bus access. Each CAN message can transmit from 0 to 8 bytes of user information. The LM3S5749 includes two CAN units.

1.4.5 System Peripherals

1.4.5.1 Programmable GPIOs (see page 255)

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

The Stellaris® GPIO module is comprised of eight physical GPIO blocks, each corresponding to an individual GPIO port. The GPIO module is FiRM-compliant (compliant to the ARM Foundation IP for Real-Time Microcontrollers specification) and supports 0-61 programmable input/output pins. The number of GPIOs available depends on the peripherals being used (see “Signal Tables” on page 733 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. Pins configured as digital inputs are Schmitt-triggered.

1.4.5.2 Four Programmable Timers (see page 303)

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

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

When configured in 32-bit mode, a timer can run as a Real-Time Clock (RTC), one-shot timer or periodic timer. 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 337)

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 LM3S5749 controller offers both single-cycle SRAM and single-cycle Flash memory.

1.4.6.1 SRAM (see page 165)

The LM3S5749 static random access memory (SRAM) controller supports 64 KB SRAM. The internal SRAM of the Stellaris® devices is located at offset 0x0000.0000 of the device memory map. To reduce the number of time-consuming read-modify-write (RMW) operations, ARM has introduced *bit-banding* technology in the new Cortex-M3 processor. With a bit-band-enabled processor, certain regions in the memory map (SRAM and peripheral space) can use address aliases to access individual bits in a single, atomic operation.

1.4.6.2 Flash (see page 166)

The LM3S5749 Flash controller supports 128 KB of flash memory. The flash is organized as a set of 1-KB blocks that can be individually erased. Erasing a block causes the entire contents of the block to be reset to all 1s. These blocks are paired into a set of 2-KB blocks that can be individually protected. The blocks can be marked as read-only or execute-only, providing different levels of code protection. Read-only blocks cannot be erased or programmed, protecting the contents of those blocks from being modified. Execute-only blocks cannot be erased or programmed, and can only be read by the controller instruction fetch mechanism, protecting the contents of those blocks from being read by either the controller or by a debugger.

1.4.6.3 ROM (see page 769)

The LM3S5749 microcontroller ships with the Stellaris® family Peripheral Driver Library conveniently preprogrammed in read-only memory (ROM). The Stellaris® Peripheral Driver Library is a royalty-free software library for controlling on-chip peripherals, and includes a boot-loader capability. The library performs both peripheral initialization and peripheral control functions, with a choice of polled or interrupt-driven peripheral support, and takes full advantage of the stellar interrupt performance of the ARM® Cortex™-M3 core. No special pragmas or custom assembly code prologue/epilogue functions are required. For applications that require in-field programmability, the royalty-free Stellaris® boot loader included in the Stellaris® Peripheral Driver Library can act as an application loader and support in-field firmware updates.

1.4.7 Additional Features

1.4.7.1 Memory Map (see page 54)

A memory map lists the location of instructions and data in memory. The memory map for the LM3S5749 controller can be found in “Memory Map” on page 54. 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 60)

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

The Luminary Micro JTAG controller works with the ARM JTAG controller built into the Cortex-M3 core. 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 72)

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.

1.4.7.4 Hibernation Module (see page 143)

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

1.4.8 Hardware Details

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

- “Pin Diagram” on page 732
- “Signal Tables” on page 733
- “Operating Characteristics” on page 748
- “Electrical Characteristics” on page 749
- “Package Information” on page 762

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:

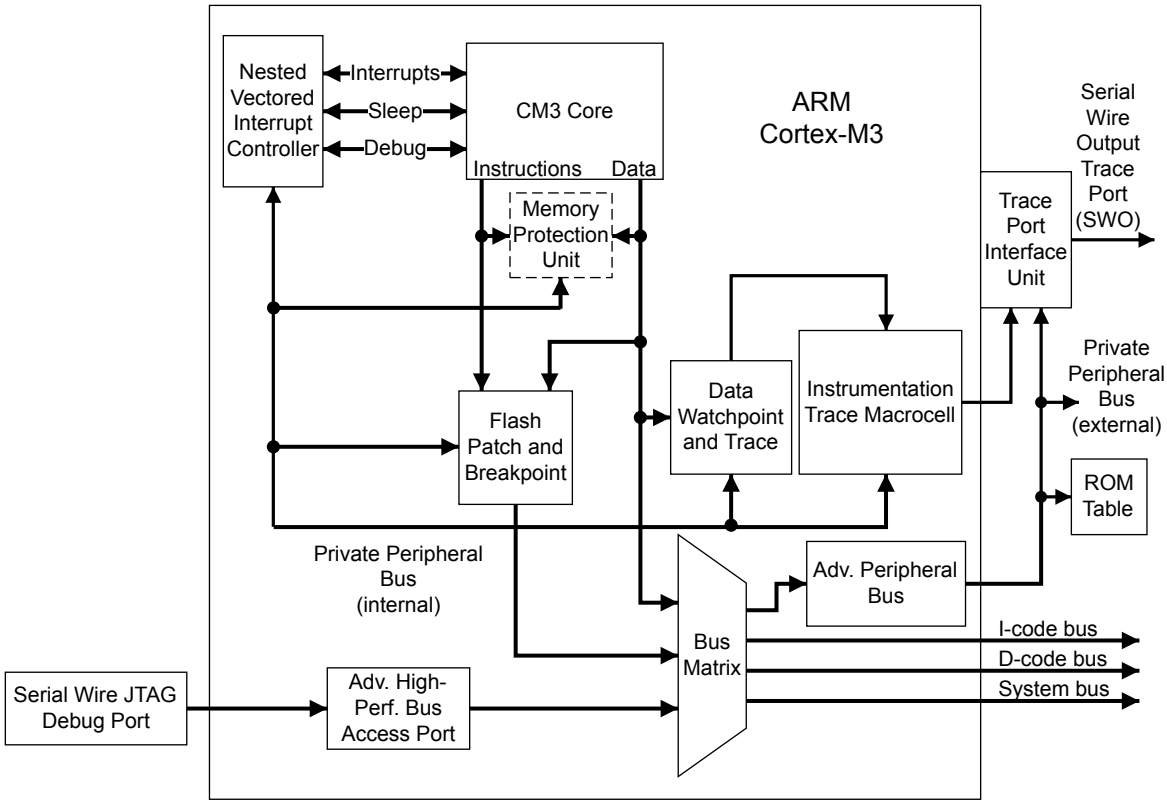
- Compact core.
- Thumb-2 instruction set, delivering the high-performance expected of an ARM core in the memory size usually associated with 8- and 16-bit devices; typically in the range of a few kilobytes of memory for microcontroller class applications.
- Rapid application execution through Harvard architecture characterized by separate buses for instruction and data.
- Exceptional interrupt handling, by implementing the register manipulations required for handling an interrupt in hardware.
- Deterministic, fast interrupt processing: always 12 cycles, or just 6 cycles with tail-chaining
- External non-maskable interrupt signal (NMI) available for immediate execution of NMI handler for safety critical applications.
- 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
  - 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
- Optimized for single-cycle flash usage
- Three sleep modes with clock gating for low power
- Single-cycle multiply instruction and hardware divide
- Atomic operations
- ARM Thumb2 mixed 16-/32-bit instruction set
- 1.25 DMIPS/MHz

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

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

2.1 Block Diagram

Figure 2-1. CPU Block Diagram



2.2 Functional Description

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

Luminary Micro has implemented the ARM Cortex-M3 core as shown in Figure 2-1 on page 49. 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.

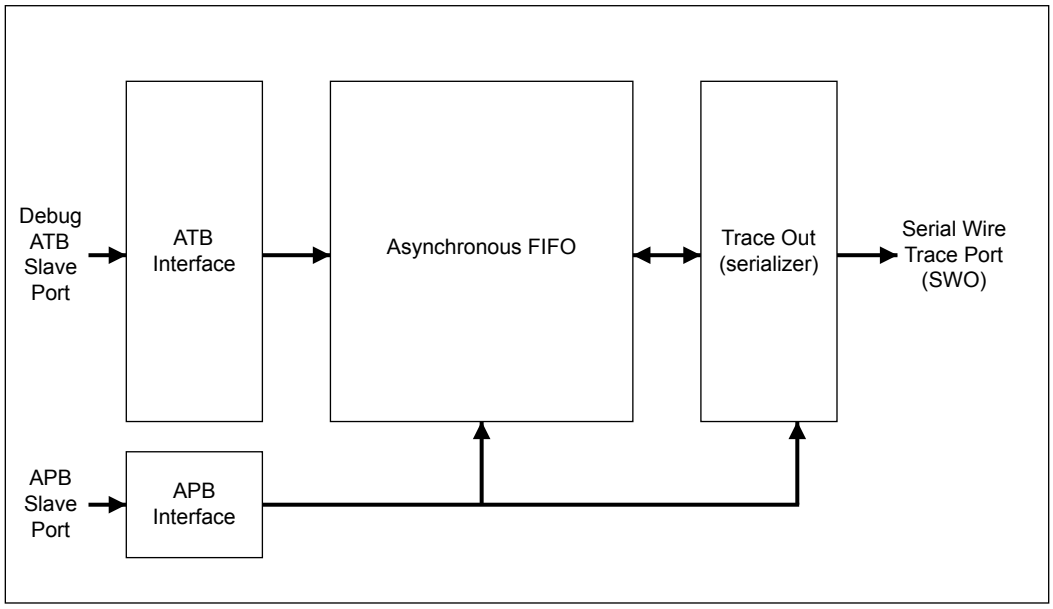
2.2.2 Embedded Trace Macrocell (ETM)

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 50. 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 LM3S5749 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.

2.2.6.1 Interrupts

The *ARM® Cortex™-M3 Technical Reference Manual* describes the maximum number of interrupts and interrupt priorities. The LM3S5749 microcontroller supports 39 interrupts with eight priority levels.

In addition to the peripheral interrupts, the system also provides for a non-maskable interrupt. The NMI is generally used in safety critical applications where the immediate execution of an interrupt handler is required. The `NMI` signal is available as an external signal so that it may be generated by external circuitry The NMI is also used internally as part of the main oscillator verification circuitry. More information on the non-maskable interrupt is located in “Non-Maskable Interrupt” on page 75.

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	Count Flag 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	Clock Source Value Description 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	Tick Interrupt Value Description 0 Counting down to 0 does not generate the interrupt request to the NVIC. Software can use the COUNTFLAG to determine if ever counted to 0. 1 Counting down to 0 pends the SysTick handler.

Bit/Field	Name	Type	Reset	Description
0	ENABLE	R/W	0	Enable Value Description 0 Counter disabled. 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.

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.
23:0	RELOAD	W1C	-	Reload 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 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 LM3S5749 controller is provided in Table 3-1 on page 54.

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*.

Table 3-1. Memory Map[a]

Start	End	Description	For details on registers, see page ...
Memory			
0x0000.0000	0x0001.FFFF	On-chip flash [b]	171
0x0002.0000	0x00FF.FFFF	Reserved	-
0x0100.0000	0x0100.2BFF	On-chip ROM	769
0x0100.2C00	0x1FFF.FFFF	Reserved	-
0x2000.0000	0x2000.FFFF	Bit-banded on-chip SRAM[c]	171
0x2001.0000	0x21FF.FFFF	Reserved	-
0x2200.0000	0x221F.FFFF	Bit-band alias of 0x2000.0000 through 0x200F.FFFF	165
0x2220.0000	0x3FFF.FFFF	Reserved	-
FiRM Peripherals			
0x4000.0000	0x4000.0FFF	Watchdog timer	340
0x4000.1000	0x4000.3FFF	Reserved	-
0x4000.4000	0x4000.4FFF	GPIO Port A	263
0x4000.5000	0x4000.5FFF	GPIO Port B	263
0x4000.6000	0x4000.6FFF	GPIO Port C	263
0x4000.7000	0x4000.7FFF	GPIO Port D	263
0x4000.8000	0x4000.8FFF	SSI0	452
0x4000.9000	0x4000.9FFF	SSI1	452
0x4000.A000	0x4000.BFFF	Reserved	-
0x4000.C000	0x4000.CFFF	UART0	404
0x4000.D000	0x4000.DFFF	UART1	404
0x4000.E000	0x4001.FFFF	Reserved	-
Peripherals			
0x4002.0000	0x4002.07FF	I2C Master 0	493
0x4002.0800	0x4002.0FFF	I2C Slave 0	506
0x4002.1000	0x4002.17FF	I2C Master 1	493
0x4002.1800	0x4002.1FFF	I2C Slave 1	506
0x4002.2000	0x4002.3FFF	Reserved	-
0x4002.4000	0x4002.4FFF	GPIO Port E	263
0x4002.5000	0x4002.5FFF	GPIO Port F	263
0x4002.6000	0x4002.6FFF	GPIO Port G	263
0x4002.7000	0x4002.7FFF	GPIO Port H	263
0x4002.8000	0x4002.8FFF	PWM	670

Start	End	Description	For details on registers, see page ...
0x4002.9000	0x4002.BFFF	Reserved	-
0x4002.C000	0x4002.CFFF	QEI0	719
0x4002.D000	0x4002.FFFF	Reserved	-
0x4003.0000	0x4003.0FFF	Timer0	314
0x4003.1000	0x4003.1FFF	Timer1	314
0x4003.2000	0x4003.2FFF	Timer2	314
0x4003.3000	0x4003.3FFF	Timer3	314
0x4003.4000	0x4003.7FFF	Reserved	-
0x4003.8000	0x4003.8FFF	ADC	369
0x4003.9000	0x4003.BFFF	Reserved	-
0x4003.C000	0x4003.CFFF	Analog Comparators	648
0x4003.D000	0x4003.FFFF	Reserved	-
0x4004.0000	0x4004.0FFF	CAN0 Controller	533
0x4004.1000	0x4004.1FFF	CAN1 Controller	533
0x4004.2000	0x4004.FFFF	Reserved	-
0x4005.0000	0x4005.0FFF	USB	576
0x4005.1000	0x4005.7FFF	Reserved	-
0x4005.8000	0x4005.8FFF	GPIO Port A (AHB aperture)	263
0x4005.9000	0x4005.9FFF	GPIO Port B (AHB aperture)	263
0x4005.A000	0x4005.AFFF	GPIO Port C (AHB aperture)	263
0x4005.B000	0x4005.BFFF	GPIO Port D (AHB aperture)	263
0x4005.C000	0x4005.CFFF	GPIO Port E (AHB aperture)	263
0x4005.D000	0x4005.DFFF	GPIO Port F (AHB aperture)	263
0x4005.E000	0x4005.EFFF	GPIO Port G (AHB aperture)	263
0x4005.F000	0x4005.FFFF	GPIO Port H (AHB aperture)	263
0x4006.0000	0x400F.BFFF	Reserved	-
0x400F.C000	0x400F.CFFF	Hibernation Module	151
0x400F.D000	0x400F.DFFF	Flash control	171
0x400F.E000	0x400F.EFFF	System control	82
0x400F.F000	0x400F.FFFF	uDMA	214
0x4010.0000	0x41FF.FFFF	Reserved	-
0x4200.0000	0x43FF.FFFF	Bit-banded alias of 0x4000.0000 through 0x400F.FFFF	-
0x4400.0000	0xDFFF.FFFF	Reserved	-
Private Peripheral Bus			
0xE000.0000	0xE000.0FFF	Instrumentation Trace Macrocell (ITM)	*ARM® Cortex™-M3 Technical Reference Manual*
0xE000.1000	0xE000.1FFF	Data Watchpoint and Trace (DWT)	*ARM® Cortex™-M3 Technical Reference Manual*

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

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 57 lists all exception types. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 39 interrupts (listed in Table 4-2 on page 58).

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

Internally, the highest user-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 position number) determines the order in which the processor activates them. For example, if both GPIO Port A and GPIO Port B are priority level 1, then GPIO Port A has higher priority.

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

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

Table 4-1. Exception Types

Exception Type	Vector Number	Priority[a]	Description
-	0	-	Stack top is loaded from 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.
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.

Exception Type	Vector Number	Priority[a]	Description
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.
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 58 lists the interrupts on the LM3S5749 controller.

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

Table 4-2. Interrupts

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Description
0-15	-	Processor exceptions
16	0	GPIO Port A
17	1	GPIO Port B
18	2	GPIO Port C
19	3	GPIO Port D
20	4	GPIO Port E
21	5	UART0
22	6	UART1
23	7	SSI0
24	8	I2C0
25	9	PWM Fault
26	10	PWM Generator 0
27	11	PWM Generator 1
28	12	PWM Generator 2
29	13	QEI0
30	14	ADC Sequence 0
31	15	ADC Sequence 1
32	16	ADC Sequence 2
33	17	ADC Sequence 3
34	18	Watchdog timer
35	19	Timer0 A
36	20	Timer0 B

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Description
37	21	Timer1 A
38	22	Timer1 B
39	23	Timer2 A
40	24	Timer2 B
41	25	Analog Comparator 0
42	26	Analog Comparator 1
43	27	Reserved
44	28	System Control
45	29	Flash Control
46	30	GPIO Port F
47	31	GPIO Port G
48	32	GPIO Port H
49	33	Reserved
50	34	SSI1
51	35	Timer3 A
52	36	Timer3 B
53	37	I2C1
54	38	Reserved
55	39	CAN0
56	40	CAN1
57-58	41-42	Reserved
59	43	Hibernation Module
60	44	USB
61	45	PWM Generator 3
62	46	uDMA Software
63	47	uDMA Error
64-70	48-54	Reserved

5 JTAG Interface

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

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

The Luminary Micro JTAG controller works with the ARM JTAG controller built into the Cortex-M3 core. 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.

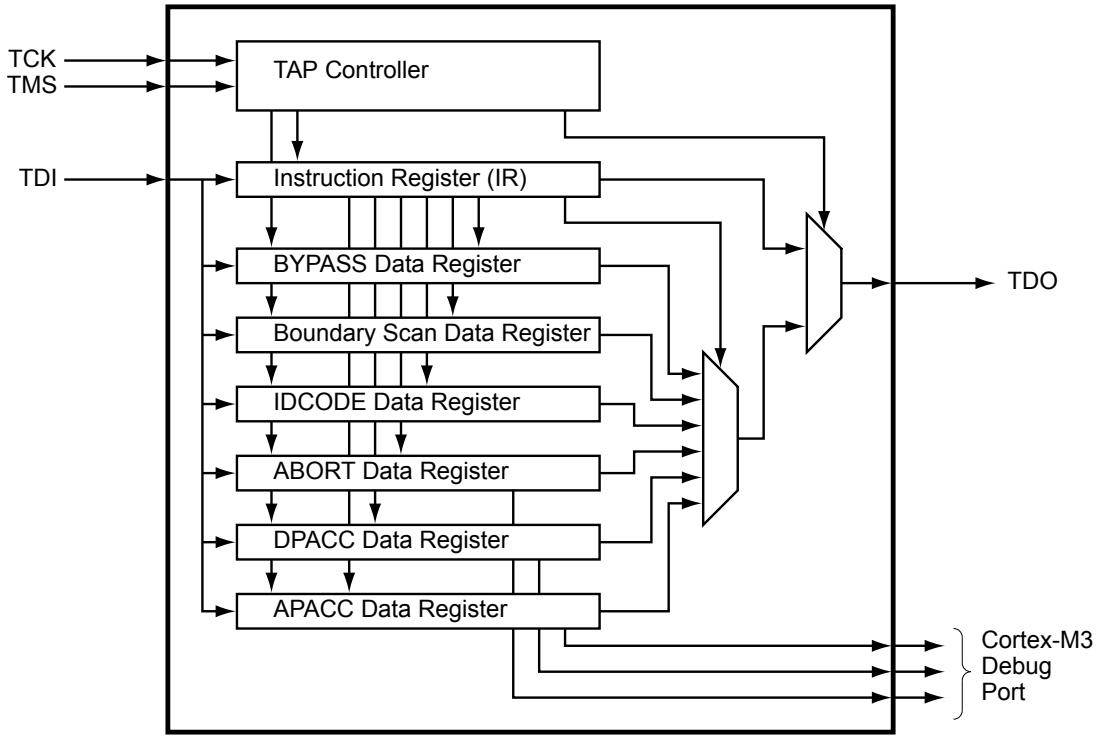
The Stellaris® JTAG module has the following features:

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

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

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

Some instructions, like EXTEST and INTEST, operate on data currently in a DR chain and do not capture, shift, or update any of the chains. Instructions that are not implemented decode to the BYPASS instruction to ensure that the serial path between TDI and TDO is always connected (see Table 5-2 on page 67 for a list of implemented instructions).

See “JTAG and Boundary Scan” on page 754 for JTAG timing diagrams.

5.2.1 JTAG Interface Pins

The JTAG interface consists of four standard pins: TCK, TMS, TDI, and TDO. These pins and their associated reset state are given in Table 5-1 on page 62. 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
TCK	Input	Enabled	Disabled	N/A	N/A
TMS	Input	Enabled	Disabled	N/A	N/A
TDI	Input	Enabled	Disabled	N/A	N/A
TDO	Output	Enabled	Disabled	2-mA driver	High-Z

5.2.1.1 Test Clock Input (TCK)

The TCK pin is the clock for the JTAG module. This clock is provided so the test logic can operate independently of any other system clocks. 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.

5.2.1.2 Test Mode Select (TMS)

The TMS pin selects the next state of the JTAG TAP controller. TMS is sampled on the rising edge of TCK. Depending on the current TAP state and the sampled value of TMS, the next state 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 module and associated registers are reset to their default values. This procedure should be performed to initialize the JTAG controller. The JTAG Test Access Port state machine can be seen in its entirety in Figure 5-2 on page 64.

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.

5.2.1.3 Test Data Input (TDI)

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

5.2.1.4 Test Data Output (TDO)

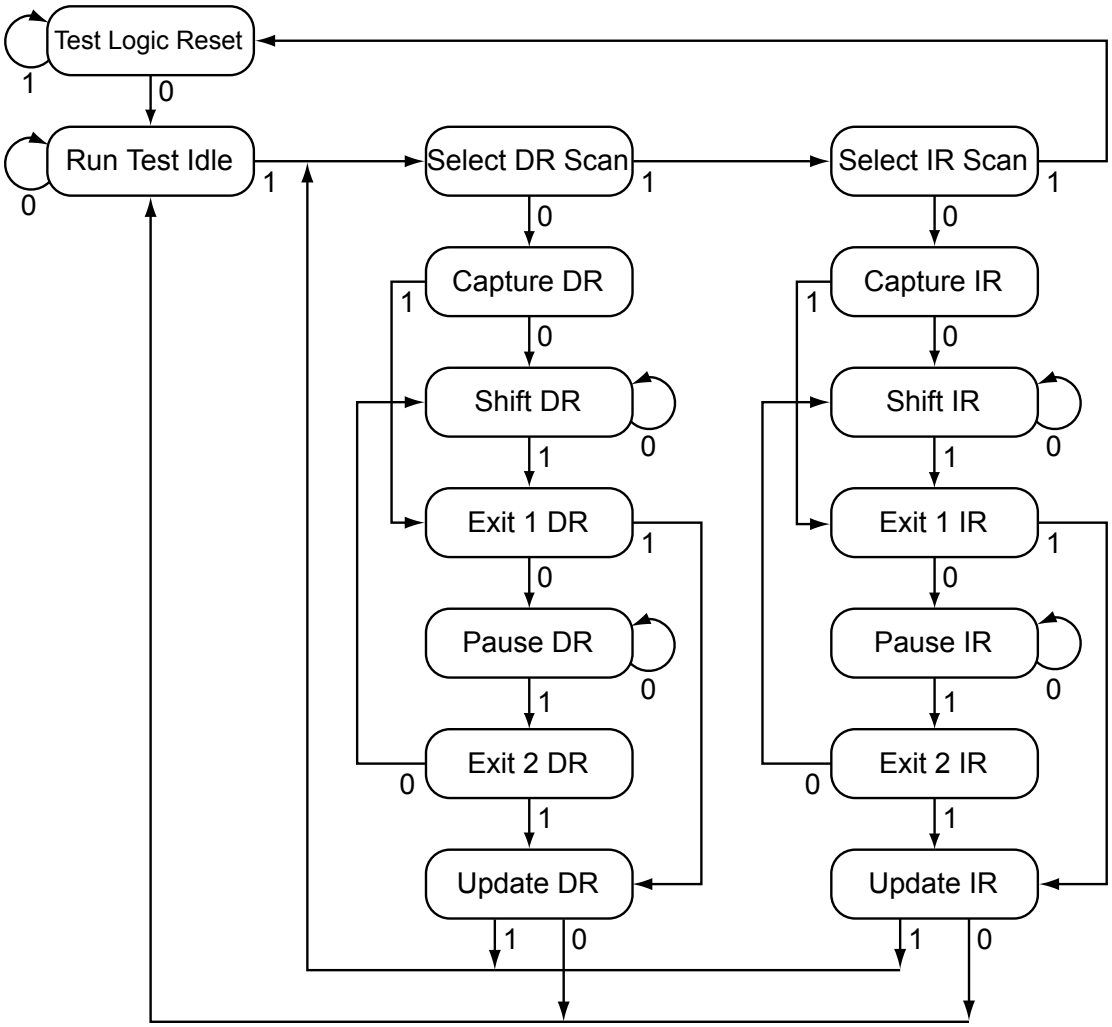
The TDO pin provides an output stream of serial information from the IR chain or the DR chains. The value of TDO depends on the current TAP state, the current instruction, and the data in the chain being accessed. In order to save power when the JTAG port is not being used, the TDO pin is placed in an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

By default, the internal pull-up resistor on the TDO pin is enabled after reset. 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 64. The TAP controller state machine is reset to the Test-Logic-Reset state on the assertion of a Power-On-Reset (POR). In order to reset the JTAG module after the device has been powered on, the TMS input must be held HIGH for five TCK clock cycles, resetting the TAP controller and all associated JTAG chains. Asserting the correct sequence on the TMS pin allows the JTAG module to shift in new instructions, shift in data, or idle during extended testing sequences. For detailed information on the function of the TAP controller and the operations that occur in each state, please refer to *IEEE Standard 1149.1*.

Figure 5-2. Test Access Port State Machine



5.2.3 Shift Registers

The Shift Registers consist of a serial shift register chain and a parallel load register. The serial shift register chain samples specific information during the TAP controller's CAPTURE states and allows this information to be shifted out 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 67.

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/SWD port pins default to their JTAG/SWD configurations. The default configuration includes enabling digital functionality (setting **GPIODEN** to 1), enabling the pull-up resistors (setting **GPIOPUR** to 1), and enabling the alternate hardware function (setting **GPIOAFSEL** to 1) for the PC[3:0] JTAG/SWD pins.

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

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

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the NMI pin (PB7) and the four JTAG/SWD pins (PC[3:0]). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 274), **GPIO Pull-Up Select (GPIOPUR)** register (see page 280), and **GPIO Digital Enable (GPIODEN)** register (see page 284) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 286) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 287) have been set to 1.

Recovering a "Locked" Device

Note: Performing the sequence below causes the nonvolatile registers discussed in “Nonvolatile Register Programming” on page 168 to be restored to their factory default values. The mass erase of the flash memory caused by the below sequence occurs prior to the nonvolatile registers being restored.

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

1. Assert and hold the RST signal.
2. Perform the JTAG-to-SWD switch sequence.
3. Perform the SWD-to-JTAG switch sequence.
4. Perform the JTAG-to-SWD switch sequence.
5. Perform the SWD-to-JTAG switch sequence.
6. Perform the JTAG-to-SWD switch sequence.
7. Perform the SWD-to-JTAG switch sequence.
8. Perform the JTAG-to-SWD switch sequence.
9. Perform the SWD-to-JTAG switch sequence.

10. Perform the JTAG-to-SWD switch sequence.
11. Perform the SWD-to-JTAG switch sequence.
12. Release the $\overline{\text{RST}}$ signal.
13. Wait 400 ms.
14. Power-cycle the device.

The JTAG-to-SWD and SWD-to-JTAG switch sequences are described in “ARM Serial Wire Debug (SWD)” on page 66. When performing switch sequences for the purpose of recovering the debug capabilities of the device, only steps 1 and 2 of the switch sequence in the section called “JTAG-to-SWD Switching” on page 66 must be performed.

5.2.4.2 ARM Serial Wire Debug (SWD)

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

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

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

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

JTAG-to-SWD Switching

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

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

SWD-to-JTAG Switching

To switch the operating mode of the Debug Access Port (DAP) from SWD to JTAG mode, the external debug hardware must send a switch sequence to the device. The 16-bit switch sequence for switching to JTAG mode is defined as b1110011100111100, transmitted LSB first. This can also be represented as 16'hE73C when transmitted LSB first. The complete switch sequence should consist of the following transactions on the TCK/SWCLK and TMS/SWDIO signals:

1. Send at least 50 TCK/SWCLK cycles with TMS/SWDIO set to 1. This ensures that both JTAG and SWD are in their reset/idle states.
2. Send the 16-bit SWD-to-JTAG switch sequence, 16'hE73C.
3. Send at least 5 TCK/SWCLK cycles with TMS/SWDIO set to 1. This ensures that if SWJ-DP was already in JTAG mode, before sending the switch sequence, the JTAG goes into the Test Logic Reset state.

5.3 Initialization and Configuration

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

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 connected between the JTAG TDI and TDO pins with a parallel load register. When the TAP Controller is placed in the correct states, bits can be shifted into the 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 67. 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.

IR[3:0]	Instruction	Description
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 is not associated with its own 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. While the EXTEST instruction is present in the Instruction Register, the Boundary Scan Data Register can be accessed to sample and shift out the current data and load new data into the Boundary Scan Data Register.

5.4.1.2 INTEST Instruction

The INTEST instruction is not associated with its own Data Register chain. 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.While the INTEXT instruction is present in the Instruction Register, the Boundary Scan Data Register can be accessed to sample and shift out the current data and load new data into the Boundary Scan Data Register.

5.4.1.3 SAMPLE/PRELOAD Instruction

The SAMPLE/PRELOAD instruction connects the Boundary Scan Data Register chain between TDI and TDO. This instruction samples the current state of the pad pins for observation and preloads new test data. Each GPIO pad has an associated input, output, and output enable signal. When the TAP controller enters the Capture DR state during this instruction, the input, output, and output-enable signals to each of the GPIO pads are captured. These samples are serially shifted out 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 70 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 71 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 71 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 70 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, or the Test-Logic-Reset state is entered. Please see “IDCODE Data Register” on page 69 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 70 for more information.

5.4.2 Data Registers

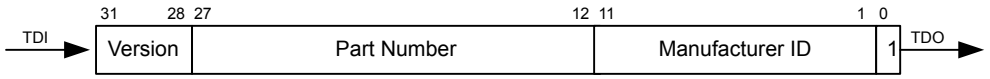
The JTAG module contains six Data Registers. These include: IDCODE, BYPASS, Boundary Scan, APACC, DPACC, and ABORT serial Data Register chains. Each of these Data Registers is discussed in the following sections.

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 70. The standard requires that every JTAG-compliant device implement either the IDCODE instruction or the BYPASS instruction as the default instruction. The LSB of the IDCODE Data Register is defined to be a 1 to distinguish it from the BYPASS instruction, which has an LSB of 0. This allows auto configuration test tools to determine which instruction is the default instruction.

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

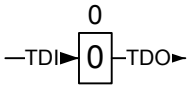
Figure 5-3. IDCODE Register Format



5.4.2.2 BYPASS Data Register

The format for the 1-bit BYPASS Data Register defined by the *IEEE Standard 1149.1* is shown in Figure 5-4 on page 70. 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

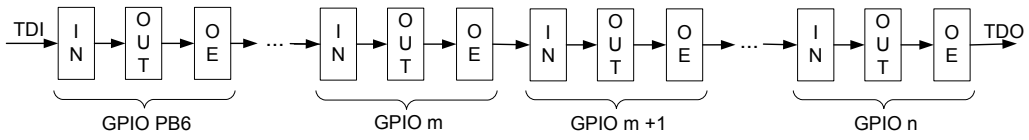


5.4.2.3 Boundary Scan Data Register

The format of the Boundary Scan Data Register is shown in Figure 5-5 on page 70. Each GPIO pin, starting with a GPIO pin next to the JTAG port pins, is included in the Boundary Scan Data Register. Each GPIO pin has three associated digital signals that are included in the chain. These signals are input, output, and output enable, and are arranged in that order as can be seen in the figure. For detailed information on the order of the input, output, and output enable bits for each of the GPIO ports and any other pins included on the Boundary Scan Data Chain, please refer to the Stellaris® Family Boundary Scan Description Language (BSDL) files, downloadable from www.luminarymicro.com.

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



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 72
- Local control, such as reset (see “Reset Control” on page 72), power (see “Power Control” on page 75) and clock control (see “Clock Control” on page 75)
- System control (Run, Sleep, and Deep-Sleep modes), see “System Control” on page 79

6.1.1 Device Identification

Several read-only registers provide software with information on the microcontroller, such as version, part number, SRAM size, flash size, and other features. See the **DID0**, **DID1**, and **DC0**-**DC7** 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 72.
2. Power-on reset (POR), see “Power-On Reset (POR)” on page 73.
3. Internal brown-out (BOR) detector, see “Brown-Out Reset (BOR)” on page 73.
4. Software-initiated reset (with the software reset registers), see “Software Reset” on page 74.
5. A watchdog timer reset condition violation, see “Watchdog Timer Reset” on page 74.
6. MOSC failure, see “Main Oscillator Verification Failure” on page 75.

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

6.1.2.2 $\overline{\text{RST}}$ 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 60). The external reset sequence is as follows:

1. The external reset pin ($\overline{\text{RST}}$) is asserted and then de-asserted.

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

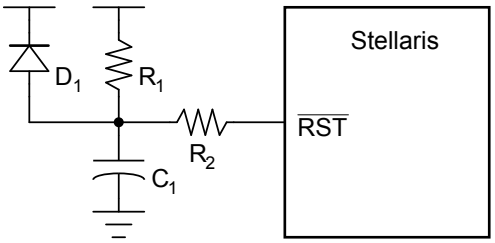
The external reset timing is shown in Figure 25-4 on page 755.

6.1.2.3 Power-On Reset (POR)

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

The device must be operating within the specified operating parameters at the point when the on-chip power-on reset pulse is complete. The 3.3-V power supply to the device must reach 3.0 V within 10 msec of it crossing 2.0 V to guarantee proper operation. For applications that require the use of an external reset to hold the device in reset longer than the internal POR, the $\overline{RST}$ input may be used with the circuit as shown in Figure 6-1 on page 73.

Figure 6-1. External Circuitry to Extend Reset



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

The Power-On Reset sequence is as follows:

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

The internal POR is only active on the initial power-up of the controller. The Power-On Reset timing is shown in Figure 25-5 on page 756.

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}). If a brown-out condition is detected, the system may generate a controller interrupt or a system reset.

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

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

The internal Brown-Out Reset timing is shown in Figure 25-6 on page 756.

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 79). 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 25-7 on page 756.

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 25-8 on page 756.

6.1.3 Non-Maskable Interrupt

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

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

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

6.1.3.1 NMI Pin

The alternate function to GPIO port pin B7 is an NMI signal. The alternate function must be enabled in the GPIO for the signal to be used as an interrupt, as described in “General-Purpose Input/Outputs (GPIOs)” on page 255. Note that enabling the NMI alternate function requires the use of the GPIO lock and commit function just like the GPIO port pins associated with JTAG/SWD functionality. The active sense of the NMI signal is High; asserting the enabled NMI signal above V_{IH} initiates the NMI interrupt sequence.

6.1.3.2 Main Oscillator Verification Failure

The main oscillator verification circuit may generate a reset event, at which time a Power-on Reset is generated and control is transferred to the NMI handler. The NMI handler is used to address the main oscillator verification failure because the necessary code can be removed from the general reset handler, speeding up reset processing. The detection circuit is enabled using the CVAL bit in the **Main Oscillator Control (MOSCCTL)** register. The main oscillator verification error is indicated in the main oscillator fail status bit (MOSCFAIL) bit in the **Reset Cause (RESC)** register. The main oscillator verification circuit action is described in more detail in “Clock Control” on page 75.

6.1.4 Power Control

The Stellaris® microcontroller provides an integrated LDO regulator that may be used to provide power to the majority of the controller's internal logic. For power reduction, 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.

Note: On the printed circuit board, use the LDO output as the source of VDD25 input. In addition, the LDO requires decoupling capacitors. See “On-Chip Low Drop-Out (LDO) Regulator Characteristics” on page 750.

6.1.5 Clock Control

System control determines the control of clocks in this part.

6.1.5.1 Fundamental Clock Sources

There are multiple clock sources for use in the 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. The internal oscillator is the clock source the device uses during and following POR.

If the main oscillator is required, software must enable the main oscillator following reset and allow the main oscillator to stabilize before changing the clock reference.

- **Main Oscillator (MOSC).** The main oscillator provides a frequency-accurate clock source by one of two means: an external single-ended clock source is connected to the OSC0 input pin, or an external crystal is connected across the OSC0 input and OSC1 output pins. If the PLL is being used, the crystal value must be one of the supported frequencies between 3.579545 MHz through 16.384 MHz (inclusive). If the PLL is not being used, the crystal may be any one of the supported frequencies between 1 MHz and 16.384 MHz. The single-ended clock source range is from DC through the specified speed of the device. The supported crystals are listed in the XTAL bit field in the **RCC** register (see page 91).

- **Internal 30-kHz Oscillator.** The internal 30-kHz oscillator is similar to the internal oscillator, except that it provides an operational frequency of 30 kHz ± 50%. It is intended for use during Deep-Sleep power-saving modes. This power-savings mode benefits from reduced internal switching and also allows the main oscillator to be powered down.

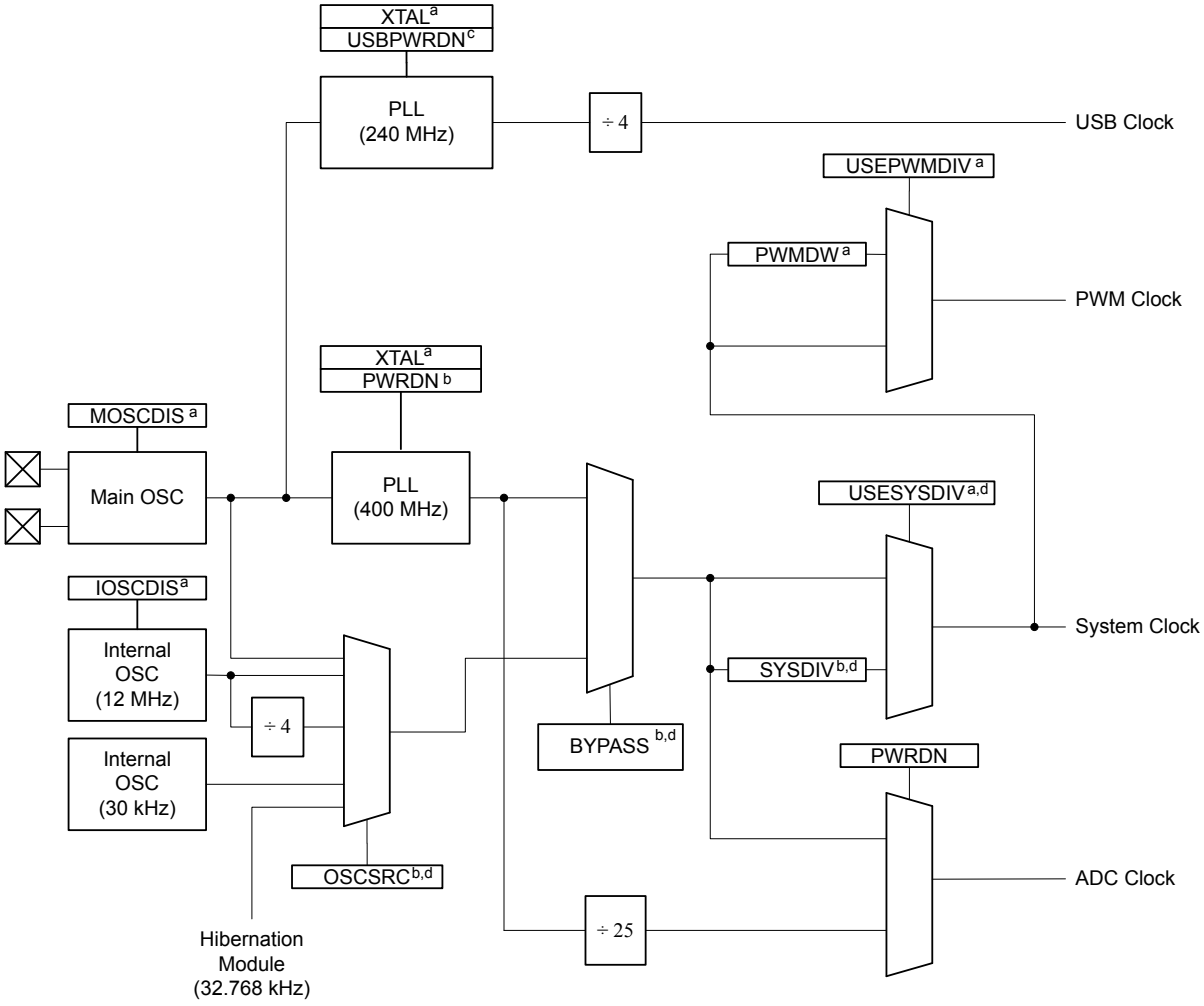
- **External Real-Time Oscillator.** The external real-time oscillator provides a low-frequency, accurate clock reference. It is intended to provide the system with a real-time clock source. The real-time oscillator is part of the Hibernation Module (see “Hibernation Module” on page 143) and may also provide an accurate source of Deep-Sleep or Hibernate mode power savings.

The internal system clock (SysClk), is derived from any of the above sources plus two others: the output of the main internal PLL, and the 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 16.384 MHz (inclusive).

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

Figure 6-2 on page 77 shows the logic for the main clock tree. The peripheral blocks are driven by the system clock signal and can be individually enabled/disabled. The ADC clock signal is automatically divided down to 16 MHz for proper ADC operation. The PWM clock signal is a synchronous divide of the system clock to provide the PWM circuit with more range (set with PWMDIV in **RCC**).

Figure 6-2. Main Clock Tree



a. Control provided by RCC register bit/field.
b. Control provided by RCC register bit/field or RCC2 register bit/field, if overridden with RCC2 register bit USERCC2.
c. Control provided by RCC2 register bit/field.
d. Also may be controlled by DSLPCLKCFG when in deep sleep mode.

Note: The figure above shows all features available on all Stellaris® DustDevil-class devices.

6.1.5.2 Crystal Configuration for the Main Oscillator (MOSC)

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

The XTAL bit in the **RCC** register (see page 91) 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.

6.1.5.3 Main PLL Frequency Configuration

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

If the main oscillator provides the clock reference to the main PLL, the translation provided by hardware and used to program the PLL is available for software in the **XTAL to PLL Translation (PLLCFG)** register (see page 96). The internal translation provides a translation within ± 1% of the targeted PLL VCO frequency.

The Crystal Value field (XTAL) on page 91 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.

To configure the external 32-kHz real-time oscillator as the PLL input reference, program the OSCRC2 field in the **Run-Mode Clock Configuration 2 (RCC2)** register to be 0x7.

6.1.5.4 USB PLL Frequency Configuration

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

6.1.5.5 PLL Modes

Both PLLs have two modes of operation: Normal and Power-Down

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

The modes are programmed using the **RCC**/**RCC2** register fields (see page 91 and page 99).

6.1.5.6 PLL Operation

If a PLL configuration is changed, the PLL output frequency is unstable until it reconverges (relocks) to the new setting. The time between the configuration change and relock is T_{READY} (see Table 25-8 on page 753) for the main PLL and $T_{USBREADY}$ for the USB PLL. During the relock time, the affected PLL is not usable as a clock reference.

Either PLL is changed by one of the following:

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

A counter is defined to measure both the T_{READY} and $T_{USBREADY}$ requirements. The counter is clocked by the main oscillator. The range of the main oscillator has been taken into account and the down counter is set to 0x1200 (that is, ~600 µs at an 8.192 MHz external oscillator clock). When the XTAL value is greater than 0x0f, the down counter is set to 0x2400 to maintain the required lock time on higher frequency crystal inputs. Hardware is provided to keep the PLL from being used as

a system clock until the T_{READY} condition is met after one of the two changes above. It is the user's responsibility to have a stable clock source (like the main oscillator) before the **RCC**/**RCC2** register is switched to use the PLL.

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

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

6.1.5.7 Main Oscillator Verification Circuit

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

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

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

6.1.6 System Control

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

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

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

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

- **Deep-Sleep Mode.** In Deep-Sleep mode, the clock frequency of the active peripherals may change (depending on the Run mode clock configuration) in addition to the processor clock being stopped. An interrupt returns the device to Run mode from one of the sleep modes; the sleep modes are entered on request from the code. Deep-Sleep mode is entered by first writing the Deep Sleep Enable bit in the ARM Cortex-M3 NVIC system control register and then executing a `WFI` instruction. Any properly configured interrupt event in the system 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**/**RCC2** register, to be determined by the `DSDIVORIDE` setting in the **DSLPCLKCFG** register, up to /16 or /64 respectively. When the Deep-Sleep exit event occurs, hardware brings the system clock back to the source and frequency it had at the onset of Deep-Sleep mode before enabling the clocks that had been stopped during the Deep-Sleep duration.

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

6.2 Initialization and Configuration

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

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

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

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

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

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

6.3 Register Map

Table 6-1 on page 81 lists the System Control registers, grouped by function. The offset listed is a hexadecimal increment to the register's address, relative to the System Control base address of 0x400F.E000.

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

Note: Additional Flash and ROM registers defined in the System Control register space are described in the "Internal Memory" on page 165.

Table 6-1. System Control Register Map

Offset	Name	Type	Reset	Description	See page
0x000	DID0	RO	-	Device Identification 0	83
0x004	DID1	RO	-	Device Identification 1	103
0x008	DC0	RO	0x00FF.003F	Device Capabilities 0	105
0x010	DC1	RO	0x0311.33FF	Device Capabilities 1	106
0x014	DC2	RO	0x030F.5133	Device Capabilities 2	108
0x018	DC3	RO	0x9FFF.8FFF	Device Capabilities 3	110
0x01C	DC4	RO	0x0000.30FF	Device Capabilities 4	112
0x020	DC5	RO	0x0F30.00FF	Device Capabilities 5	113
0x024	DC6	RO	0x0000.0002	Device Capabilities 6	115
0x028	DC7	RO	0x03C0.0F3F	Device Capabilities 7	116
0x030	PBORCTL	R/W	0x0000.7FFD	Brown-Out Reset Control	85
0x034	LDOPCTL	R/W	0x0000.0000	LDO Power Control	86
0x040	SRCR0	R/W	0x00000000	Software Reset Control 0	139
0x044	SRCR1	R/W	0x00000000	Software Reset Control 1	140
0x048	SRCR2	R/W	0x00000000	Software Reset Control 2	142
0x050	RIS	RO	0x0000.0000	Raw Interrupt Status	87
0x054	IMC	R/W	0x0000.0000	Interrupt Mask Control	88
0x058	MISC	R/W1C	0x0000.0000	Masked Interrupt Status and Clear	89
0x05C	RESC	R/W	-	Reset Cause	90
0x060	RCC	R/W	0x078E.3AD1	Run-Mode Clock Configuration	91
0x064	PLLCFG	RO	-	XTAL to PLL Translation	96
0x06C	GPIOHSCTL	R/W	0x0000.0000	GPIO High-Speed Control	97
0x070	RCC2	R/W	0x0780.6810	Run-Mode Clock Configuration 2	99
0x07C	MOSCCTL	R/W	0x0000.0000	Main Oscillator Control	101

Offset	Name	Type	Reset	Description	See page
0x100	RCGC0	R/W	0x00000040	Run Mode Clock Gating Control Register 0	118
0x104	RCGC1	R/W	0x00000000	Run Mode Clock Gating Control Register 1	124
0x108	RCGC2	R/W	0x00000000	Run Mode Clock Gating Control Register 2	133
0x110	SCGC0	R/W	0x00000040	Sleep Mode Clock Gating Control Register 0	120
0x114	SCGC1	R/W	0x00000000	Sleep Mode Clock Gating Control Register 1	127
0x118	SCGC2	R/W	0x00000000	Sleep Mode Clock Gating Control Register 2	135
0x120	DCGC0	R/W	0x00000040	Deep Sleep Mode Clock Gating Control Register 0	122
0x124	DCGC1	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 1	130
0x128	DCGC2	R/W	0x00000000	Deep Sleep Mode Clock Gating Control Register 2	137
0x144	DSLPCLKCFG	R/W	0x0780.0000	Deep Sleep Clock Configuration	102

6.4 Register Descriptions

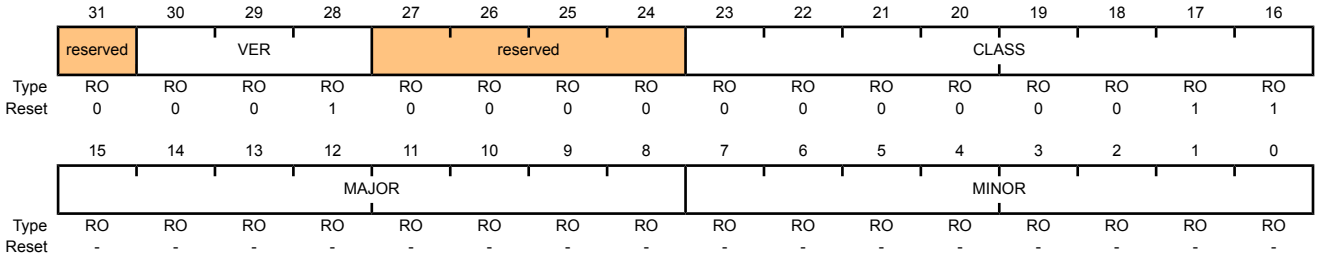
All addresses given are relative to the System Control base address of 0x400F.E000.

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 -



Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30:28	VER	RO	0x1	DID0 Version This field defines the **DID0** register format version. The version number is numeric. The value of the VER field is encoded as follows: Value Description 0x1 Second version of the **DID0** register format.
27:24	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
23:16	CLASS	RO	0x3	Device Class The CLASS field value identifies the internal design from which all mask sets are generated for all devices in a particular product line. The CLASS field value is changed for new product lines, for changes in fab process (for example, a remap or shrink), or any case where the MAJOR or MINOR fields require differentiation from prior devices. The value of the CLASS field is encoded as follows (all other encodings are reserved): Value Description 0x3 Stellaris® DustDevil-class devices

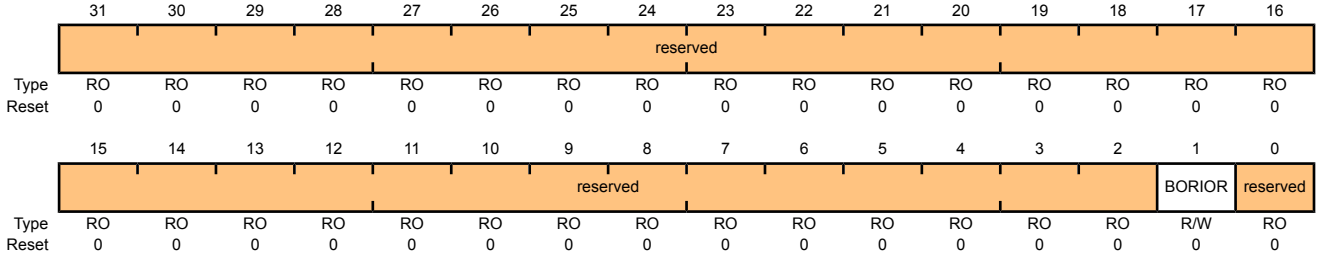
Bit/Field	Name	Type	Reset	Description
15:8	MAJOR	RO	-	Major Revision This field specifies the major revision number of the device. The major revision reflects changes to base layers of the design. The major revision number is indicated in the part number as a letter (A for first revision, B for second, 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.
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: Brown-Out Reset Control (PBORCTL), offset 0x030

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

Brown-Out Reset Control (PBORCTL)

Base 0x400F.E000
Offset 0x030
Type R/W, reset 0x0000.7FFD



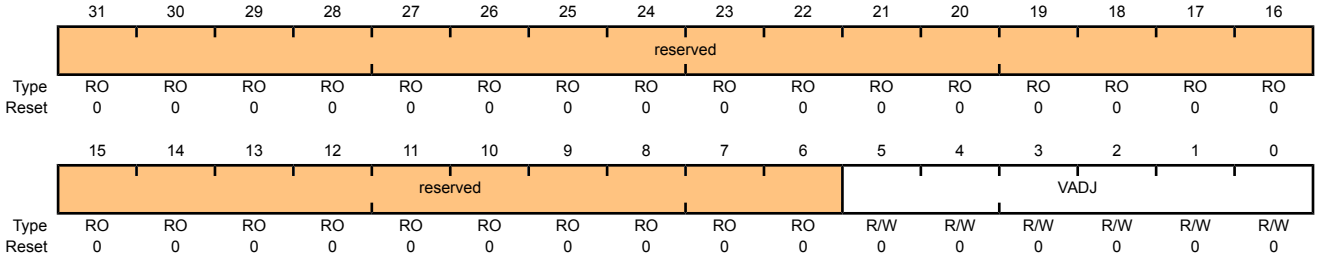
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORIOR	R/W	0	BOR Interrupt or Reset This bit controls how a BOR event is signaled to the controller. If set, a reset is signaled. Otherwise, an interrupt is signaled.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

The 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



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.

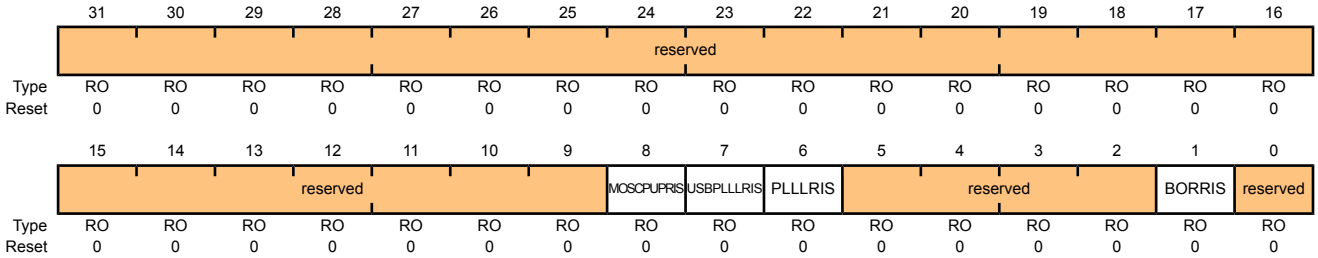
Value	V_{OUT} (V)
0x00	2.50
0x01	2.45
0x02	2.40
0x03	2.35
0x04	2.30
0x05	2.25
0x06-0x3F	Reserved
0x1B	2.75
0x1C	2.70
0x1D	2.65
0x1E	2.60
0x1F	2.55

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



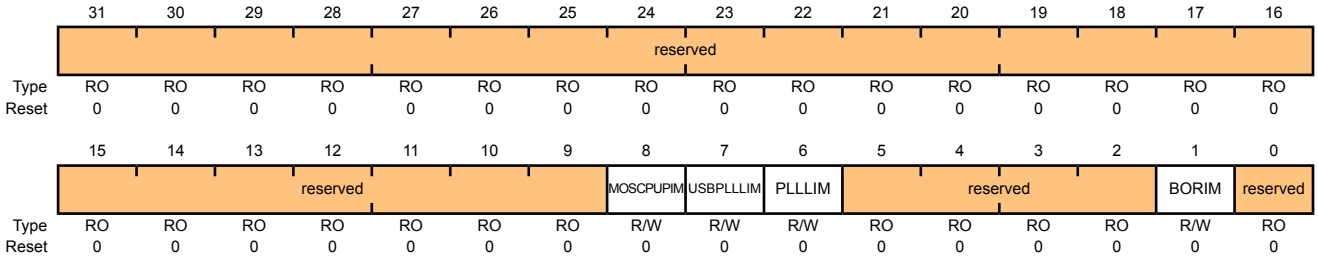
Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	MOSCPUPRIS	RO	0	MOSC Power Up Raw Interrupt Status This bit is set when the PLL $T_{MOSCPUP}$ Timer asserts.
7	USBPLLLRIS	RO	0	USB PLL Lock Raw Interrupt Status This bit is set when the USB PLL $T_{USBREADY}$ Timer asserts.
6	PLLLRIS	RO	0	PLL Lock Raw Interrupt Status This bit is set when the PLL T_{READY} Timer asserts.
5:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORRIS	RO	0	Brown-Out Reset Raw Interrupt Status This bit is the raw interrupt status for any brown-out conditions. If set, a brown-out condition is currently active. This is an unregistered signal from the brown-out detection circuit. An interrupt is reported if the `BORIM` bit in the **IMC** register is set and the `BORIOR` bit in the **PBORCTL** register is cleared.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

Central location for system control interrupt masks.

Interrupt Mask Control (IMC)

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



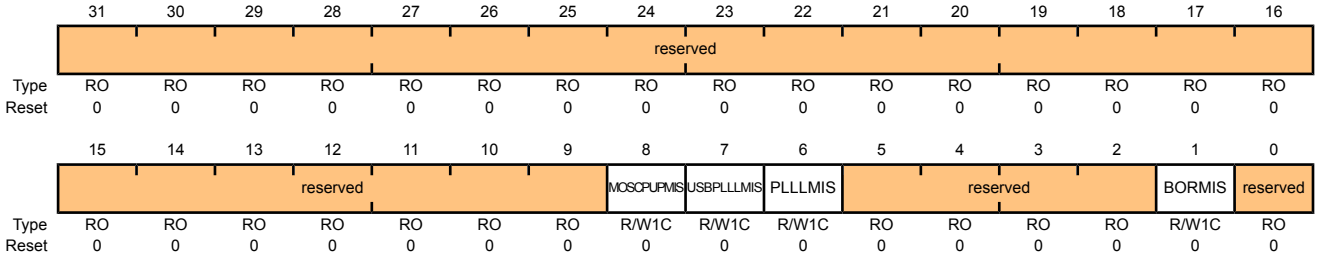
Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	MOSCPUPIM	R/W	0	MOSC Power Up Interrupt Mask This bit specifies whether a MOSC power up intterupt is promoted to a controller interrupt. If set, an interrupt is generated if `MOSCPUPRIS` in **RIS** is set; otherwise, an interrupt is not generated.
7	USBPLLLIM	R/W	0	USB PLL Lock Interrupt Mask This bit specifies whether a USB PLL Lock interrupt is promoted to a controller interrupt. If set, an interrupt is generated if `USBPLLLRIS` in **RIS** is set; otherwise, an interrupt is not generated.
6	PLLLIM	R/W	0	PLL Lock Interrupt Mask This bit specifies whether a PLL Lock interrupt is promoted to a controller interrupt. If set, an interrupt is generated if `PLLLRIS` in **RIS** is set; otherwise, an interrupt is not generated.
5:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORIM	R/W	0	Brown-Out Reset Interrupt Mask This bit specifies whether a brown-out condition is promoted to a controller interrupt. If set, an interrupt is generated if `BORRIS` is set; otherwise, an interrupt is not generated.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

On a read, this register gives the current masked status value of the corresponding interrupt. All of the bits are R/W1C and this action also clears the corresponding raw interrupt bit in the **RIS** register (see page 87).

Masked Interrupt Status and Clear (MISC)

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



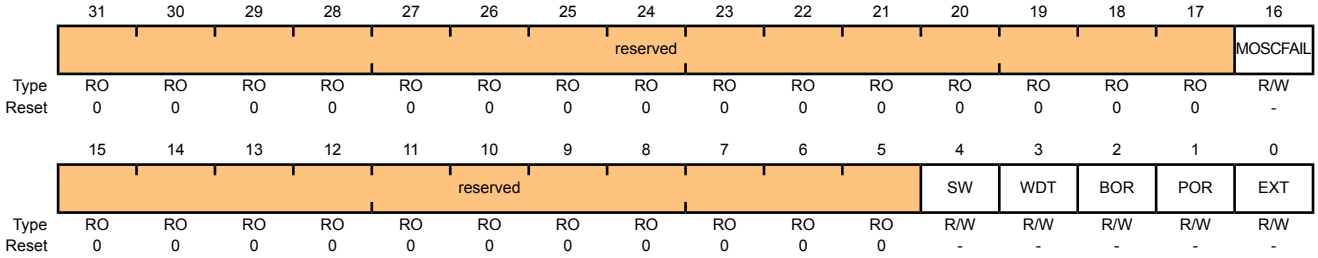
Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	MOSCPUPMIS	R/W1C	0	MOSC Power Up Masked Interrupt Status This bit is set when the $T_{MOSCPUP}$ timer asserts. The interrupt is cleared by writing a 1 to this bit.
7	USBPLLLMIS	R/W1C	0	USB PLL Lock Masked Interrupt Status This bit is set when the USB PLL $T_{USBREADY}$ timer asserts. The interrupt is cleared by writing a 1 to this bit.
6	PLLLMIS	R/W1C	0	PLL Lock Masked Interrupt Status This bit is set when the PLL T_{READY} timer asserts. The interrupt is cleared by writing a 1 to this bit.
5:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	BORMIS	R/W1C	0	BOR Masked Interrupt Status The `BORMIS` is simply the `BORRIS` ANDed with the mask value, `BORIM`.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

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

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

Reset Cause (RESC)

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



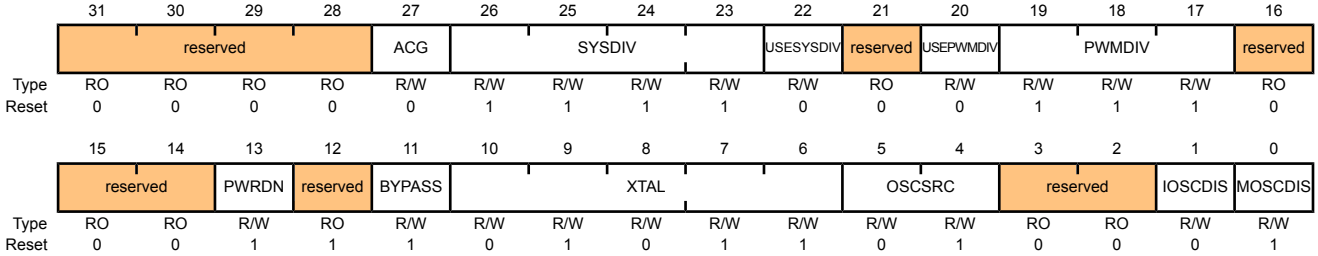
Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	MOSCFAIL	R/W	-	MOSC Failure Reset When set, indicates the MOSC circuit was enable for clock validation and failed. This generated a reset event.
15:5	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	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 ($\overline{\text{RST}}$ 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 0x078E.3AD1



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 400 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 91), 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	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
20	USEPWMDIV	R/W	0	Enable PWM Clock Divisor Use the PWM clock divider as the source for the PWM clock.

Bit/Field	Name	Type	Reset	Description
19:17	PWMDIV	R/W	0x7	PWM Unit Clock Divisor This field specifies the binary divisor used to predivide the system clock down for use as the timing reference for the PWM module. This clock is only power 2 divide and rising edge is synchronous without phase shift from the system clock. Value Divisor 0x0 /2 0x1 /4 0x2 /8 0x3 /16 0x4 /32 0x5 /64 0x6 /64 0x7 /64 (default)
16:14	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.
12	reserved	RO	1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	BYPASS	R/W	1	PLL Bypass 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. **Note:** The ADC must be clocked from the PLL or directly from a 14-MHz to 18-MHz clock source to operate properly. While the ADC works in a 14-18 MHz range, to maintain a 1 M sample/second rate, the ADC must be provided a 16-MHz clock source.

Bit/Field	Name	Type	Reset	Description
10:6	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.

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

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

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

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

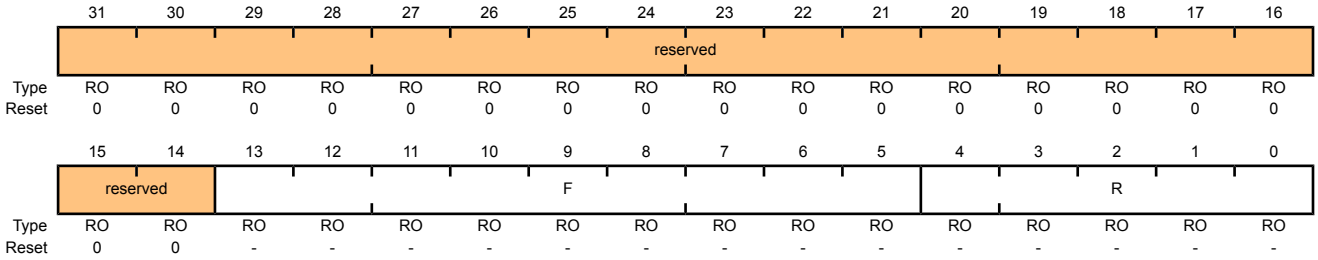
This register provides a means of translating external crystal frequencies into the appropriate PLL settings. This register is initialized during the reset sequence and updated anytime that the XTAL field changes in the **Run-Mode Clock Configuration (RCC)** register (see page 91).

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

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

XTAL to PLL Translation (PLLCFG)

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



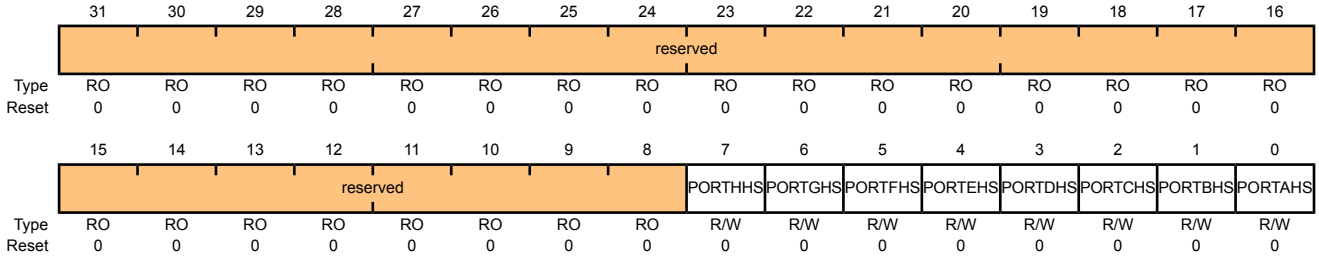
Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13:5	F	RO	-	PLL F Value This field specifies the value 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: GPIO High-Speed Control (GPIOHSCTL), offset 0x06C

This register provides the user the ability to change the GPIO ports to run on a single-cycle bus equivalent to the processor clock instead of the legacy bus with two-cycle access. The address aperture in the memory map will change for the ports that are enabled for high-speed access (see Table 10-3 on page 262).

GPIO High-Speed Control (GPIOHSCTL)

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



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	PORTHHS	R/W	0	Port H High-Speed When set, the memory aperture for Port H is selected to be high speed (single-cycle). Otherwise, the legacy aperture (two-cycle) is chosen.
6	PORTGHS	R/W	0	Port G High-Speed When set, the memory aperture for Port G is selected to be high speed (single-cycle). Otherwise, the legacy aperture (two-cycle) is chosen.
5	PORTFHS	R/W	0	Port F High-Speed When set, the memory aperture for Port F is selected to be high speed (single-cycle). Otherwise, the legacy aperture (two-cycle) is chosen.
4	PORTEHS	R/W	0	Port E High-Speed When set, the memory aperture for Port E is selected to be high speed (single-cycle). Otherwise, the legacy aperture (two-cycle) is chosen.
3	PORTDHS	R/W	0	Port D High-Speed When set, the memory aperture for Port D is selected to be high speed (single-cycle). Otherwise, the legacy aperture (two-cycle) is chosen.
2	PORTCHS	R/W	0	Port C High-Speed When set, the memory aperture for Port C is selected to be high speed (single-cycle). Otherwise, the legacy aperture (two-cycle) is chosen.
1	PORTBHS	R/W	0	Port B High-Speed When set, the memory aperture for Port B is selected to be high speed (single-cycle). Otherwise, the legacy aperture (two-cycle) is chosen.

Bit/Field	Name	Type	Reset	Description
0	PORTAHS	R/W	0	Port A High-Speed When set, the memory aperture for Port A is selected to be high speed (single-cycle). Otherwise, the legacy aperture (two-cycle) is chosen.

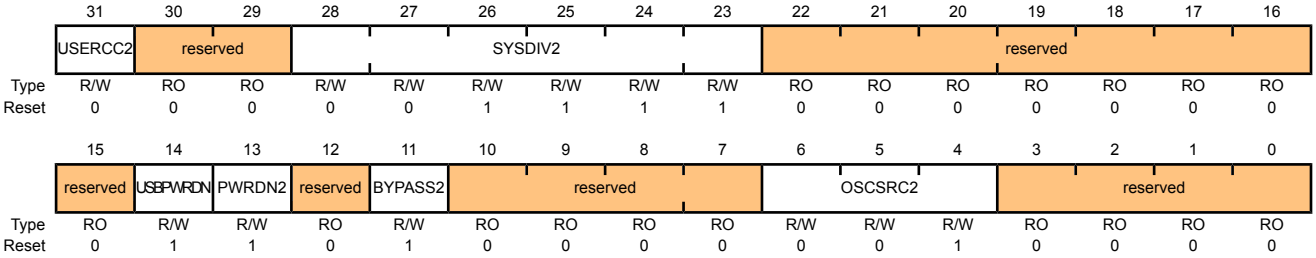
Register 11: Run-Mode Clock Configuration 2 (RCC2), offset 0x070

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

The SYSDIV2 field is 2 bits wider than the SYSDIV field in the **RCC** register so that additional larger divisors are possible, allowing a lower system clock frequency for improved Deep Sleep power consumption. The PLL VCO frequency is 400 MHz.

Run-Mode Clock Configuration 2 (RCC2)

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



Bit/Field	Name	Type	Reset	Description
31	USERCC2	R/W	0	Use RCC2 When set, overrides the **RCC** register fields.
30:29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28:23	SYSDIV2	R/W	0x0F	System Clock Divisor Specifies which divisor is used to generate the system clock from the PLL output. This field is wider than the **RCC** register SYSDIV field in order to provide additional divisor values. This permits the system clock to be run at much lower frequencies during Deep Sleep mode. For example, where the **RCC** register SYSDIV encoding of 1111 provides /16, the **RCC2** register SYSDIV2 encoding of 111111 provides /64.
22:15	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	USBPWRDN	R/W	1	Power-Down USB PLL When set, powers down the USB PLL.
13	PWRDN2	R/W	1	Power-Down PLL When set, powers down the PLL.
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

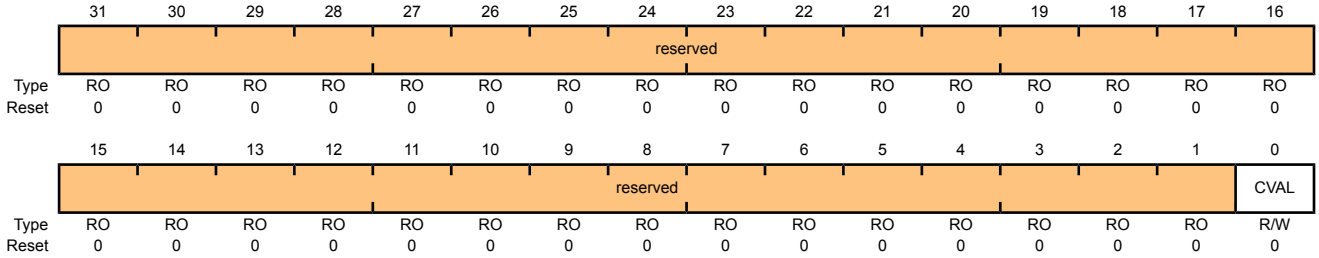
Bit/Field	Name	Type	Reset	Description
11	BYPASS2	R/W	1	Bypass PLL When set, bypasses the PLL for the clock source.
10:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:4	OSCSRC2	R/W	0x1	Oscillator Source Selects the input source for the OSC. The values are: Value Description 0x0 MOSC Main oscillator 0x1 IOSC Internal oscillator 0x2 IOSC/4 Internal oscillator / 4 0x3 30 kHz 30-kHz internal oscillator 0x4 Reserved 0x5 Reserved 0x6 Reserved 0x7 32 kHz 32.768-kHz external oscillator
3:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 12: Main Oscillator Control (MOSCCTL), offset 0x07C

This register provides control over the features of the main oscillator, including the ability to enable the MOSC clock validation circuit. When enabled, this circuit monitors the energy on the MOSC pins to provide a Clock Valid signal. If the clock goes invalid after being enabled, the part does a hardware reset and reboots to the NMI handler.

Main Oscillator Control (MOSCCTL)

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



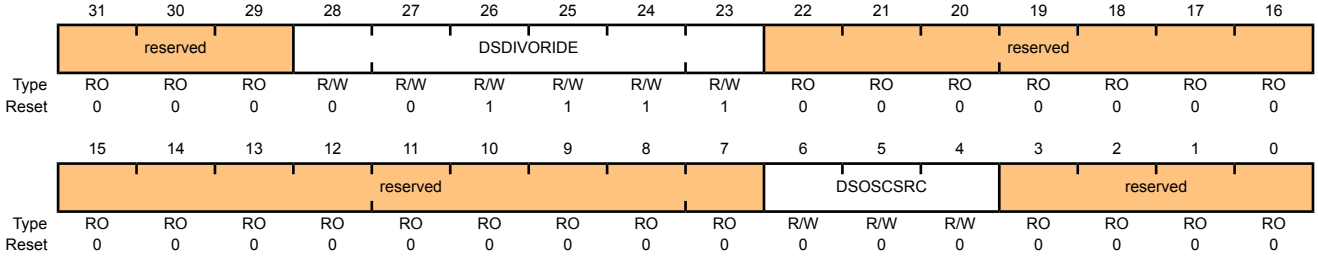
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	CVAL	R/W	0	Clock Validation for MOSC When set, the monitor circuit is enabled.

Register 13: Deep Sleep Clock Configuration (DSLPCLKCFG), offset 0x144

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

Deep Sleep Clock Configuration (DSLPCLKCFG)

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



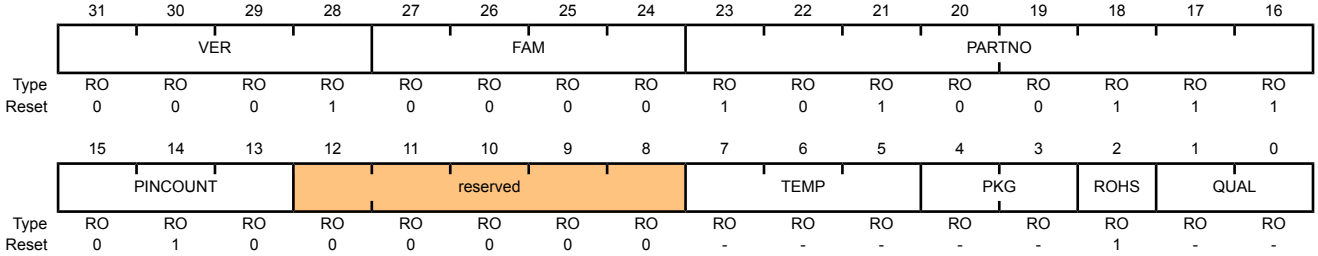
Bit/Field	Name	Type	Reset	Description
31:29	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28:23	DSDIVORIDE	R/W	0x0F	Divider Field Override 6-bit system divider field to override when Deep-Sleep occurs with PLL running.
22:7	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:4	DSOSCSRC	R/W	0x0	Clock Source Specifies the clock source during Deep-Sleep mode. Value Description 0x0 MOSC Use main oscillator as source. 0x1 IOSC Use internal 12-MHz oscillator as source. 0x2 Reserved 0x3 30 kHz Use 30-kHz internal oscillator as source. 0x4 Reserved 0x5 Reserved 0x6 Reserved 0x7 32 kHz Use 32.768-kHz external oscillator as source.
3:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 14: Device Identification 1 (DID1), offset 0x004

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

Device Identification 1 (DID1)

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



Bit/Field	Name	Type	Reset	Description
31:28	VER	RO	0x1	DID1 Version This field defines the **DID1** register format version. The version number is numeric. The value of the VER field is encoded as follows (all other encodings are reserved): Value Description 0x1 Second version of the **DID1** register format.
27:24	FAM	RO	0x0	Family This field provides the family identification of the device within the Luminary Micro product portfolio. The value is encoded as follows (all other encodings are reserved): Value Description 0x0 Stellaris family of microcontollers, that is, all devices with external part numbers starting with LM3S.
23:16	PARTNO	RO	0xA7	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 0xA7 LM3S5749
15:13	PINCOUNT	RO	0x2	Package Pin Count This field specifies the number of pins on the device package. The value is encoded as follows (all other encodings are reserved): Value Description 0x2 100-pin package

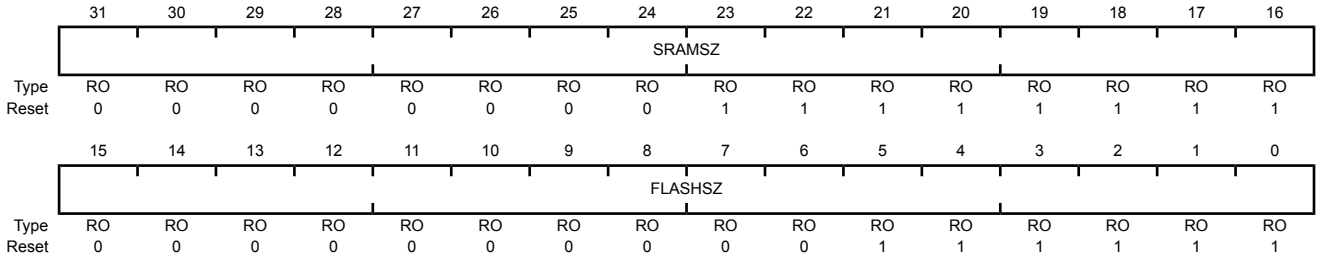
Bit/Field	Name	Type	Reset	Description
12:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:5	TEMP	RO	-	Temperature Range This field specifies the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

Register 15: Device Capabilities 0 (DC0), offset 0x008

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

Device Capabilities 0 (DC0)

Base 0x400F.E000
Offset 0x008
Type RO, reset 0x00FF.003F



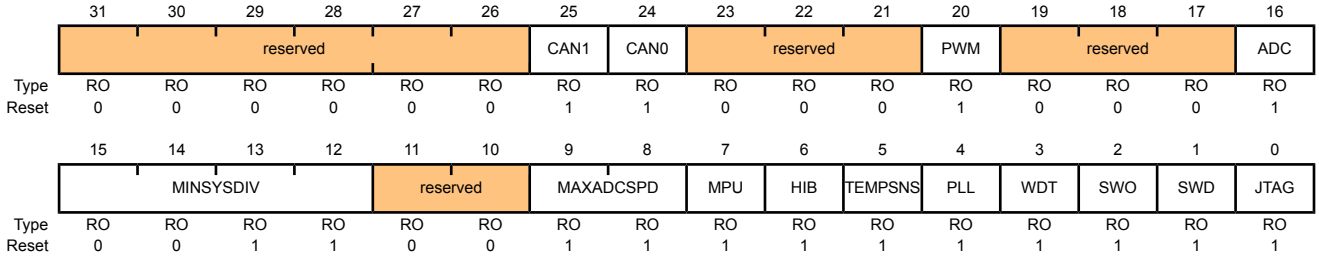
Bit/Field	Name	Type	Reset	Description
31:16	SRAMSZ	RO	0x00FF	SRAM Size Indicates the size of the on-chip SRAM memory. Value Description 0x00FF 64 KB of SRAM
15:0	FLASHSZ	RO	0x003F	Flash Size Indicates the size of the on-chip flash memory. Value Description 0x003F 128 KB of Flash

Register 16: Device Capabilities 1 (DC1), offset 0x010

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

Device Capabilities 1 (DC1)

Base 0x400F.E000
Offset 0x010
Type RO, reset 0x0311.33FF



Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	CAN1	RO	1	CAN Module 1 Present When set, indicates that CAN unit 1 is present.
24	CAN0	RO	1	CAN Module 0 Present When set, indicates that CAN unit 0 is present.
23:21	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
20	PWM	RO	1	PWM Module Present When set, indicates that the PWM module is present.
19:17	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	ADC	RO	1	ADC Module Present. When set, indicates that the ADC module is present.
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.
11:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

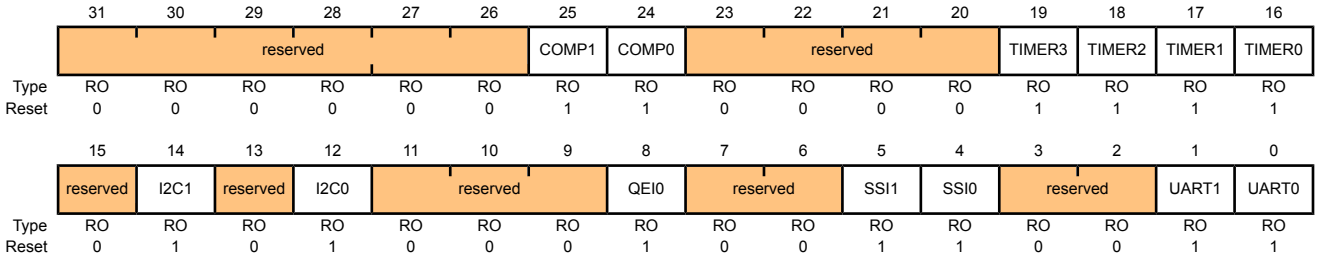
Bit/Field	Name	Type	Reset	Description
9:8	MAXADCSPD	RO	0x3	Max ADC Speed. This field indicates the maximum rate at which the ADC samples data. Value Description 0x3 1M samples/second
7	MPU	RO	1	MPU Present. When set, indicates that the Cortex-M3 Memory Protection Unit (MPU) module is present. See the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

Register 17: Device Capabilities 2 (DC2), offset 0x014

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

Device Capabilities 2 (DC2)

Base 0x400F.E000
Offset 0x014
Type RO, reset 0x030F.5133



Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	COMP1	RO	1	Analog Comparator 1 Present. When set, indicates that analog comparator 1 is present.
24	COMP0	RO	1	Analog Comparator 0 Present. When set, indicates that analog comparator 0 is present.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	RO	1	Timer 3 Present. When set, indicates that General-Purpose Timer module 3 is present.
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	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	RO	1	I2C Module 1 Present. When set, indicates that I2C module 1 is present.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	RO	1	I2C Module 0 Present. When set, indicates that I2C module 0 is present.
11:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
8	QEI0	RO	1	QEI0 Present. When set, indicates that QEI module 0 is present.
7:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	RO	1	SSI1 Present. When set, indicates that SSI module 1 is present.
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 18: Device Capabilities 3 (DC3), offset 0x018

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

Device Capabilities 3 (DC3)

Base 0x400F.E000
Offset 0x018
Type RO, reset 0x9FFF.8FFF

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	32KHZ	reserved		CCP4	CCP3	CCP2	CCP1	CCP0	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PWMFAULT	reserved			C1O	C1PLUS	C1MINUS	C0O	C0PLUS	C0MINUS	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31	32KHZ	RO	1	32KHz Input Clock Available. When set, indicates an even CCP pin is present and can be used as a 32-KHz input clock.
30:29	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
28	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	ADC7	RO	1	ADC7 Pin Present. When set, indicates that ADC pin 7 is present.
22	ADC6	RO	1	ADC6 Pin Present. When set, indicates that ADC pin 6 is present.
21	ADC5	RO	1	ADC5 Pin Present. When set, indicates that ADC pin 5 is present.
20	ADC4	RO	1	ADC4 Pin Present. When set, indicates that ADC pin 4 is present.
19	ADC3	RO	1	ADC3 Pin Present. When set, indicates that ADC pin 3 is present.
18	ADC2	RO	1	ADC2 Pin Present. When set, indicates that ADC pin 2 is present.
17	ADC1	RO	1	ADC1 Pin Present. When set, indicates that ADC pin 1 is present.
16	ADC0	RO	1	ADC0 Pin Present. When set, indicates that ADC pin 0 is present.
15	PWMFAULT	RO	1	PWM Fault Pin Present. When set, indicates that a PWM Fault pin is present. See **DC5** for specific Fault pins on this device.

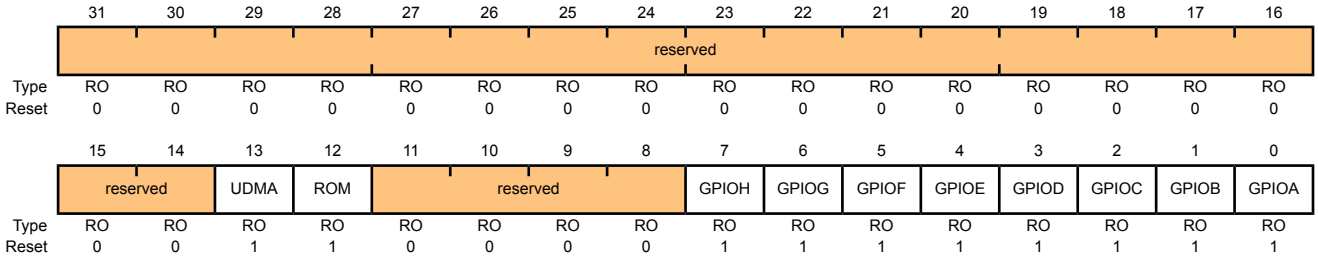
Bit/Field	Name	Type	Reset	Description
14:12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	C1O	RO	1	C1o Pin Present. When set, indicates that the analog comparator 1 output pin is present.
10	C1PLUS	RO	1	C1+ Pin Present. When set, indicates that the analog comparator 1 (+) input pin is present.
9	C1MINUS	RO	1	C1- Pin Present. When set, indicates that the analog comparator 1 (-) input pin is present.
8	C0O	RO	1	C0o Pin Present. When set, indicates that the analog comparator 0 output pin is present.
7	C0PLUS	RO	1	C0+ Pin Present. When set, indicates that the analog comparator 0 (+) input pin is present.
6	C0MINUS	RO	1	C0- Pin Present. When set, indicates that the analog comparator 0 (-) input pin is present.
5	PWM5	RO	1	PWM5 Pin Present. When set, indicates that the PWM pin 5 is present.
4	PWM4	RO	1	PWM4 Pin Present. When set, indicates that the PWM pin 4 is present.
3	PWM3	RO	1	PWM3 Pin Present. When set, indicates that the PWM pin 3 is present.
2	PWM2	RO	1	PWM2 Pin Present. When set, indicates that the PWM pin 2 is present.
1	PWM1	RO	1	PWM1 Pin Present. When set, indicates that the PWM pin 1 is present.
0	PWM0	RO	1	PWM0 Pin Present. When set, indicates that the PWM pin 0 is present.

Register 19: Device Capabilities 4 (DC4), offset 0x01C

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

Device Capabilities 4 (DC4)

Base 0x400F.E000
Offset 0x01C
Type RO, reset 0x0000.30FF



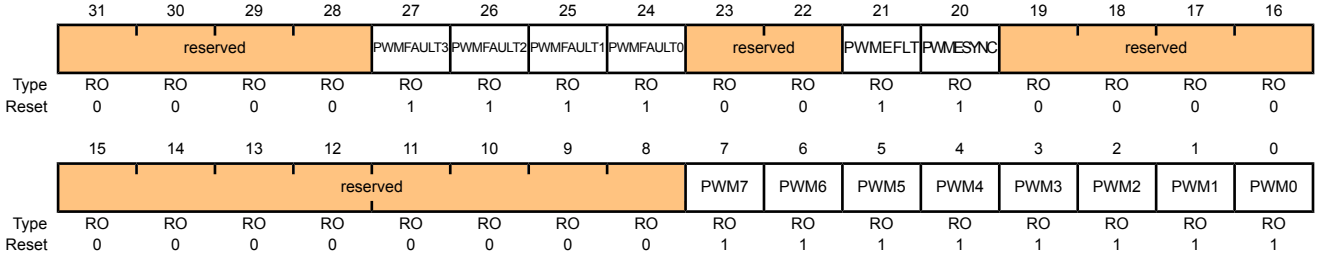
Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	RO	1	Micro-DMA is present
12	ROM	RO	1	Internal Code ROM is present
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	GPIOH	RO	1	GPIO Port H Present. When set, indicates that GPIO Port H is present.
6	GPIOG	RO	1	GPIO Port G Present. When set, indicates that GPIO Port G is present.
5	GPIOF	RO	1	GPIO Port F Present. When set, indicates that GPIO Port F is present.
4	GPIOE	RO	1	GPIO Port E Present. When set, indicates that GPIO Port E is present.
3	GPIOD	RO	1	GPIO Port D Present. When set, indicates that GPIO Port D is present.
2	GPIOC	RO	1	GPIO Port C Present. When set, indicates that GPIO Port C is present.
1	GPIOB	RO	1	GPIO Port B Present. When set, indicates that GPIO Port B is present.
0	GPIOA	RO	1	GPIO Port A Present. When set, indicates that GPIO Port A is present.

Register 20: Device Capabilities 5 (DC5), offset 0x020

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

Device Capabilities 5 (DC5)

Base 0x400F.E000
Offset 0x020
Type RO, reset 0x0F30.00FF



Bit/Field	Name	Type	Reset	Description
31:28	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
27	PWMFAULT3	RO	1	PWM Fault 3 Pin Present. When set, indicates that the PWM Fault 3 pin is present.
26	PWMFAULT2	RO	1	PWM Fault 2 Pin Present. When set, indicates that the PWM Fault 2 pin is present.
25	PWMFAULT1	RO	1	PWM Fault 1 Pin Present. When set, indicates that the PWM Fault 1 pin is present.
24	PWMFAULT0	RO	1	PWM Fault 0 Pin Present. When set, indicates that the PWM Fault 0 pin is present.
23:22	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
21	PWMEFLT	RO	1	PWM Extended Fault feature is active
20	PWMESYNC	RO	1	PWM Extended SYNC feature is active
19:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	PWM7	RO	1	PWM7 Pin Present. When set, indicates that the PWM pin 7 is present.
6	PWM6	RO	1	PWM6 Pin Present. When set, indicates that the PWM pin 6 is present.
5	PWM5	RO	1	PWM5 Pin Present. When set, indicates that the PWM pin 5 is present.
4	PWM4	RO	1	PWM4 Pin Present. When set, indicates that the PWM pin 4 is present.
3	PWM3	RO	1	PWM3 Pin Present. When set, indicates that the PWM pin 3 is present.
2	PWM2	RO	1	PWM2 Pin Present. When set, indicates that the PWM pin 2 is present.
1	PWM1	RO	1	PWM1 Pin Present. When set, indicates that the PWM pin 1 is present.

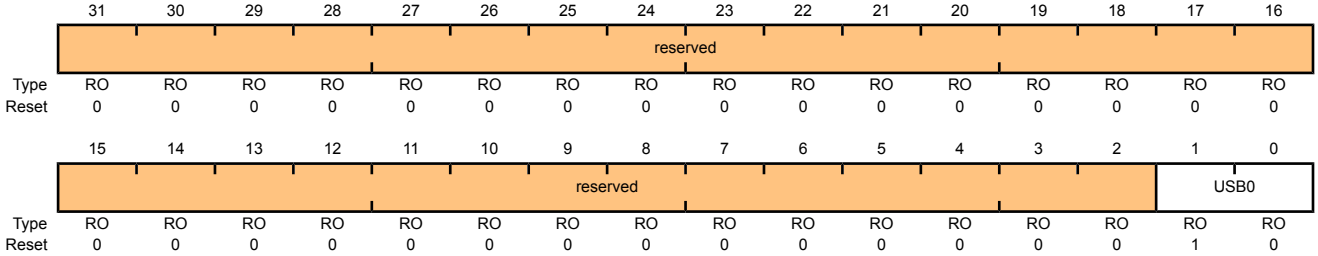
Bit/Field	Name	Type	Reset	Description
0	PWM0	RO	1	PWM0 Pin Present. When set, indicates that the PWM pin 0 is present.

Register 21: Device Capabilities 6 (DC6), offset 0x024

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

Device Capabilities 6 (DC6)

Base 0x400F.E000
Offset 0x024
Type RO, reset 0x0000.0002



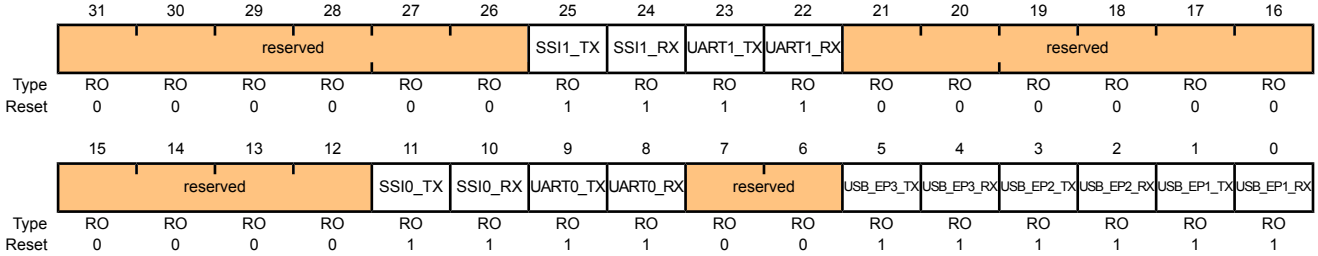
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1:0	USB0	RO	0x2	This specifies that USB0 is present and its capability Value Description 0x2 USB is Device or Host.

Register 22: Device Capabilities 7 (DC7), offset 0x028

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

Device Capabilities 7 (DC7)

Base 0x400F.E000
Offset 0x028
Type RO, reset 0x03C0.0F3F



Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	SSI1_TX	RO	1	SSI1 TX on uDMA Ch25. When set, indicates uDMA channel 25 is available and connected to the transmit path of SSI module 1.
24	SSI1_RX	RO	1	SSI1 RX on uDMA Ch24. When set, indicates uDMA channel 24 is available and connected to the receive path of SSI module 1.
23	UART1_TX	RO	1	UART1 TX on uDMA Ch23. When set, indicates uDMA channel 23 is available and connected to the transmit path of UART module 1.
22	UART1_RX	RO	1	UART1 RX on uDMA Ch22. When set, indicates uDMA channel 22 is available and connected to the receive path of UART module 1.
21:12	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	SSI0_TX	RO	1	SSI0 TX on uDMA Ch11. When set, indicates uDMA channel 11 is available and connected to the transmit path of SSI module 0.
10	SSI0_RX	RO	1	SSI0 RX on uDMA Ch10. When set, indicates uDMA channel 10 is available and connected to the receive path of SSI module 0.
9	UART0_TX	RO	1	UART0 TX on uDMA Ch9. When set, indicates uDMA channel 9 is available and connected to the transmit path of UART module 0.
8	UART0_RX	RO	1	UART0 RX on uDMA Ch8. When set, indicates uDMA channel 8 is available and connected to the receive path of UART module 0.
7:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	USB_EP3_TX	RO	1	USB EP3 TX on uDMA Ch5. When set, indicates uDMA channel 5 is available and connected to the transmit path of USB endpoint 3.
4	USB_EP3_RX	RO	1	USB EP3 RX on uDMA Ch4. When set, indicates uDMA channel 4 is available and connected to the receive path of USB endpoint 2.

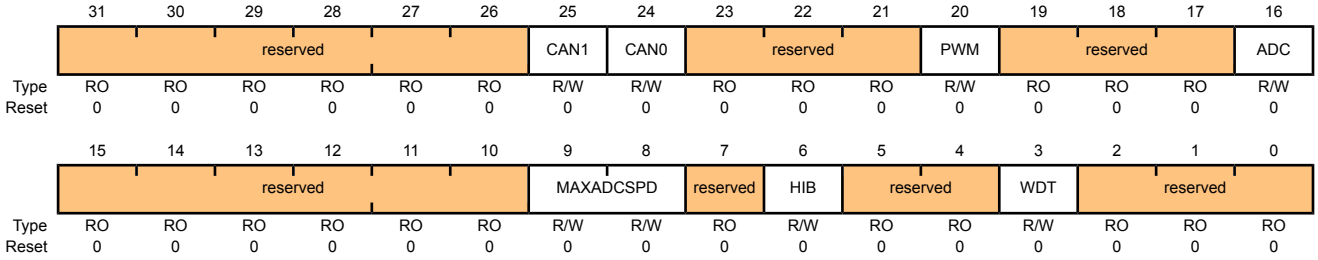
Bit/Field	Name	Type	Reset	Description
3	USB_EP2_TX	RO	1	USB EP2 TX on uDMA Ch3. When set, indicates uDMA channel 3 is available and connected to the transmit path of USB endpoint 2.
2	USB_EP2_RX	RO	1	USB EP2 RX on uDMA Ch2. When set, indicates uDMA channel 1 is available and connected to the receive path of USB endpoint 2.
1	USB_EP1_TX	RO	1	USB EP1 TX on uDMA Ch1. When set, indicates uDMA channel 1 is available and connected to the transmit path of USB endpoint 1.
0	USB_EP1_RX	RO	1	USB EP1 RX on uDMA Ch0. When set, indicates uDMA channel 0 is available and connected to the receive path of USB endpoint 1.

Register 23: 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



Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	CAN1	R/W	0	CAN1 Clock Gating Control. This bit controls the clock gating for CAN unit 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.
24	CAN0	R/W	0	CAN0 Clock Gating Control. This bit controls the clock gating for CAN unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.
23:21	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
20	PWM	R/W	0	PWM Clock Gating Control. This bit controls the clock gating for the PWM module. If set, the 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.
19:17	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	ADC	R/W	0	ADC0 Clock Gating Control. This bit controls the clock gating for SAR ADC module 0. 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.

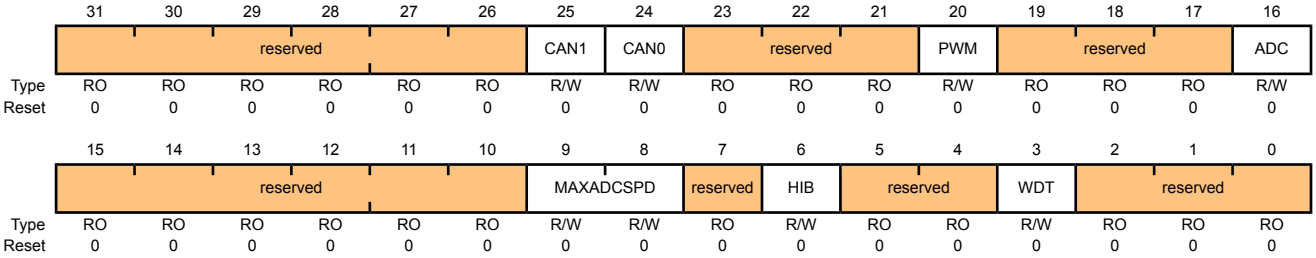
Bit/Field	Name	Type	Reset	Description
15:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	MAXADCSPD	R/W	0	ADC Sample Speed. This field sets the rate at which the ADC samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADCSPD bit as follows: Value Description 0x3 1M samples/second 0x2 500K samples/second 0x1 250K samples/second 0x0 125K samples/second
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	HIB	R/W	0	HIB Clock Gating Control. This bit controls the clock gating for the Hibernation module. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT	R/W	0	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 24: 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



Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	CAN1	R/W	0	CAN1 Clock Gating Control. This bit controls the clock gating for CAN unit 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.
24	CAN0	R/W	0	CAN0 Clock Gating Control. This bit controls the clock gating for CAN unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.
23:21	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
20	PWM	R/W	0	PWM Clock Gating Control. This bit controls the clock gating for the PWM module. If set, the 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.
19:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

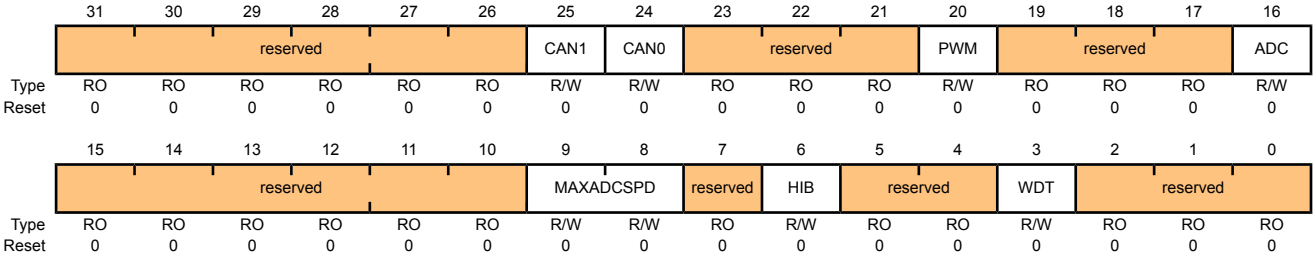
Bit/Field	Name	Type	Reset	Description
16	ADC	R/W	0	ADC0 Clock Gating Control. This bit controls the clock gating for general SAR ADC module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, a read or write to the unit generates a bus fault.
15:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	MAXADCSPD	R/W	0	ADC Sample Speed. This field sets the rate at which the ADC samples data. You cannot set the rate higher than the maximum rate.You can set the sample rate by setting the MAXADCSPD bit as follows: Value Description 0x3 1M samples/second 0x2 500K samples/second 0x1 250K samples/second 0x0 125K samples/second
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	HIB	R/W	0	HIB Clock Gating Control. This bit controls the clock gating for the Hibernation module. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT	R/W	0	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 25: 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



Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	CAN1	R/W	0	CAN1 Clock Gating Control. This bit controls the clock gating for CAN unit 1. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.
24	CAN0	R/W	0	CAN0 Clock Gating Control. This bit controls the clock gating for CAN unit 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.
23:21	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
20	PWM	R/W	0	PWM Clock Gating Control. This bit controls the clock gating for the PWM module. If set, the 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.
19:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

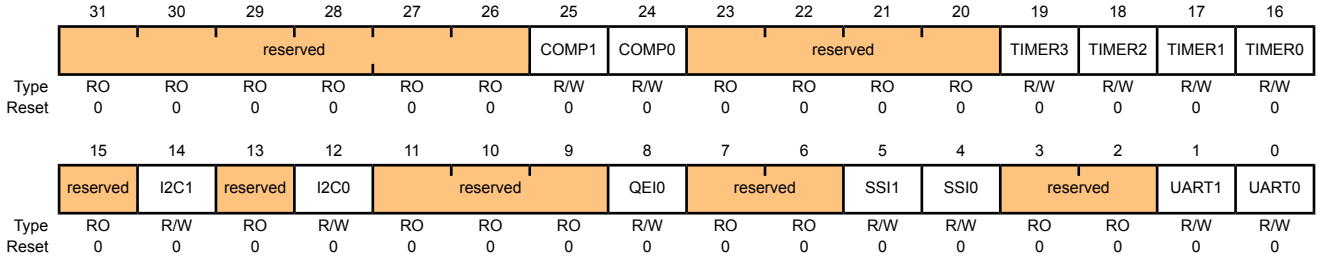
Bit/Field	Name	Type	Reset	Description
16	ADC	R/W	0	ADC0 Clock Gating Control. This bit controls the clock gating for general SAR ADC module 0. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, a read or write to the unit generates a bus fault.
15:10	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	MAXADCSPD	R/W	0	ADC Sample Speed. This field sets the rate at which the ADC samples data. You cannot set the rate higher than the maximum rate. You can set the sample rate by setting the MAXADCSPD bit as follows: Value Description 0x3 1M samples/second 0x2 500K samples/second 0x1 250K samples/second 0x0 125K samples/second
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	HIB	R/W	0	HIB Clock Gating Control. This bit controls the clock gating for the Hibernation module. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT	R/W	0	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 26: 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



Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	COMP1	R/W	0	Analog Comparator 1 Clock Gating. This bit controls the clock gating for analog comparator 1. If set, the 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.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 3. If set, the 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.
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.

Bit/Field	Name	Type	Reset	Description
17	TIMER1	R/W	0	Timer 1 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 1. If set, the 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	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	I2C1 Clock Gating Control. This bit controls the clock gating for I2C module 1. If set, the 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.
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.
11:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	QEI0	R/W	0	QEI0 Clock Gating Control. This bit controls the clock gating for QEI 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.
7:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	R/W	0	SSI1 Clock Gating Control. This bit controls the clock gating for SSI module 1. If set, the 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.
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.

Bit/Field	Name	Type	Reset	Description
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 27: 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						COMP1	COMP0	reserved				TIMER3	TIMER2	TIMER1	TIMER0
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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved	I2C1	reserved	I2C0	reserved			QEI0	reserved		SSI1	SSI0	reserved		UART1	UART0
Type	RO	R/W	RO	R/W	RO	RO	RO	R/W	RO	RO	R/W	R/W	RO	RO	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	COMP1	R/W	0	Analog Comparator 1 Clock Gating. This bit controls the clock gating for analog comparator 1. If set, the 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.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 3. If set, the 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.
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.

Bit/Field	Name	Type	Reset	Description
17	TIMER1	R/W	0	Timer 1 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 1. If set, the 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	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	I2C1 Clock Gating Control. This bit controls the clock gating for I2C module 1. If set, the 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.
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.
11:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	QEI0	R/W	0	QEI0 Clock Gating Control. This bit controls the clock gating for QEI 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.
7:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	R/W	0	SSI1 Clock Gating Control. This bit controls the clock gating for SSI module 1. If set, the 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.
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.

Bit/Field	Name	Type	Reset	Description
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 28: 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						COMP1	COMP0	reserved				TIMER3	TIMER2	TIMER1	TIMER0
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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved	I2C1	reserved	I2C0	reserved			QEI0	reserved		SSI1	SSI0	reserved		UART1	UART0
Type RO	R/W	RO	R/W	RO	RO	RO	R/W	RO	RO	R/W	R/W	RO	RO	R/W	R/W
Reset 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	COMP1	R/W	0	Analog Comparator 1 Clock Gating. This bit controls the clock gating for analog comparator 1. If set, the 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.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 3. If set, the 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.
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.

Bit/Field	Name	Type	Reset	Description
17	TIMER1	R/W	0	Timer 1 Clock Gating Control. This bit controls the clock gating for General-Purpose Timer module 1. If set, the 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	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	I2C1 Clock Gating Control. This bit controls the clock gating for I2C module 1. If set, the 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.
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.
11:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	QEI0	R/W	0	QEI0 Clock Gating Control. This bit controls the clock gating for QEI 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.
7:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	R/W	0	SSI1 Clock Gating Control. This bit controls the clock gating for SSI module 1. If set, the 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.
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.

Bit/Field	Name	Type	Reset	Description
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 29: 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															USB0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		UDMA	reserved					GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	R/W	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
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	USB0	R/W	0	USB0 Clock Gating Control. This bit controls the clock gating for Port H. 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.
15:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	R/W	0	UDMA Clock Gating Control. This bit controls the clock gating for Port H. 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.
12:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	GPIOH	R/W	0	Port H Clock Gating Control. This bit controls the clock gating for Port H. 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
6	GPIOG	R/W	0	Port G Clock Gating Control. This bit controls the clock gating for Port G. 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.
5	GPIOF	R/W	0	Port F Clock Gating Control. This bit controls the clock gating for Port F. 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.
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.
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 30: 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															USB0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		UDMA	reserved					GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	R/W	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
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	USB0	R/W	0	USB0 Clock Gating Control. This bit controls the clock gating for Port H. 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.
15:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	R/W	0	UDMA Clock Gating Control. This bit controls the clock gating for Port H. 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.
12:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	GPIOH	R/W	0	Port H Clock Gating Control. This bit controls the clock gating for Port H. 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.

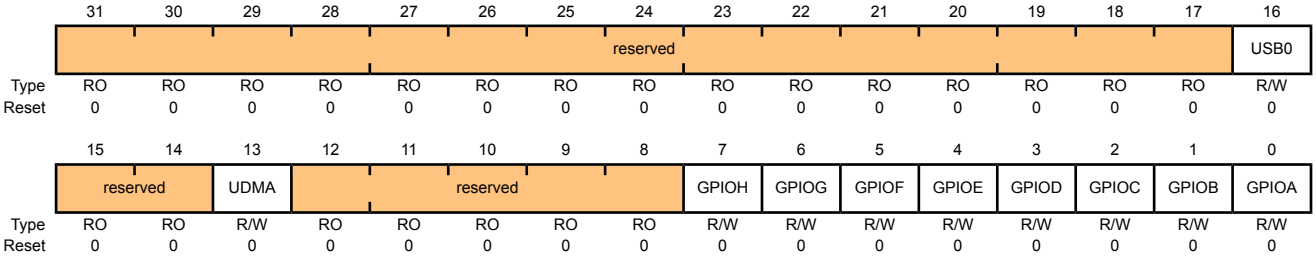
Bit/Field	Name	Type	Reset	Description
6	GPIOG	R/W	0	Port G Clock Gating Control. This bit controls the clock gating for Port G. If set, the unit receives a clock and functions. Otherwise, the unit is unclocked and disabled. If the unit is unclocked, reads or writes to the unit will generate a bus fault.
5	GPIOF	R/W	0	Port F Clock Gating Control. This bit controls the clock gating for Port F. 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.
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.
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 31: 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



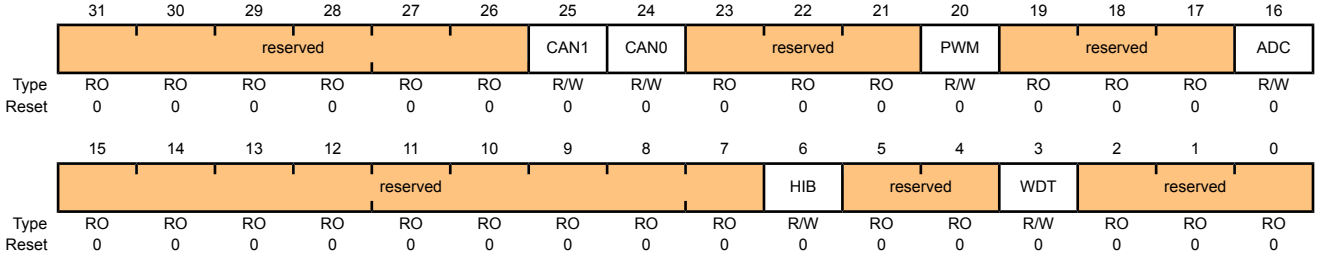
Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	USB0	R/W	0	USB0 Clock Gating Control. This bit controls the clock gating for Port H. 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.
15:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	R/W	0	UDMA Clock Gating Control. This bit controls the clock gating for Port H. 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.
12:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	GPIOH	R/W	0	Port H Clock Gating Control. This bit controls the clock gating for Port H. 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
6	GPIOG	R/W	0	Port G Clock Gating Control. This bit controls the clock gating for Port G. 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.
5	GPIOF	R/W	0	Port F Clock Gating Control. This bit controls the clock gating for Port F. 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.
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.
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 32: 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



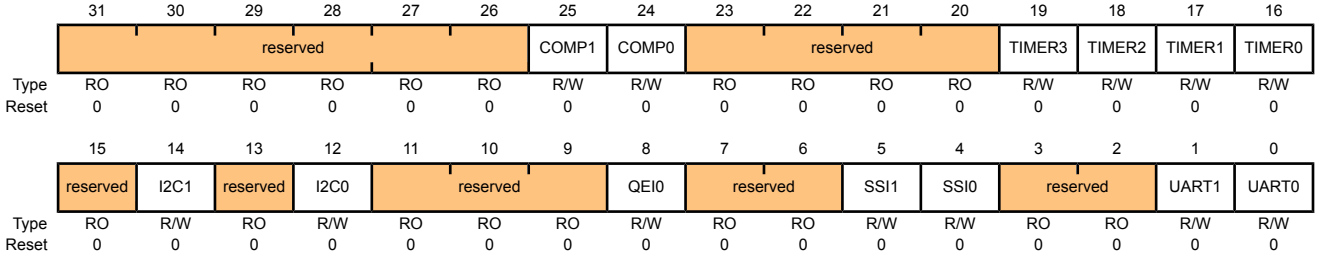
Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	CAN1	R/W	0	CAN1 Reset Control. Reset control for CAN unit 1.
24	CAN0	R/W	0	CAN0 Reset Control. Reset control for CAN unit 0.
23:21	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
20	PWM	R/W	0	PWM Reset Control. Reset control for PWM module.
19:17	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	ADC	R/W	0	ADC0 Reset Control. Reset control for SAR ADC module 0.
15:7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	HIB	R/W	0	HIB Reset Control. Reset control for the Hibernation module.
5:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	WDT	R/W	0	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 33: 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



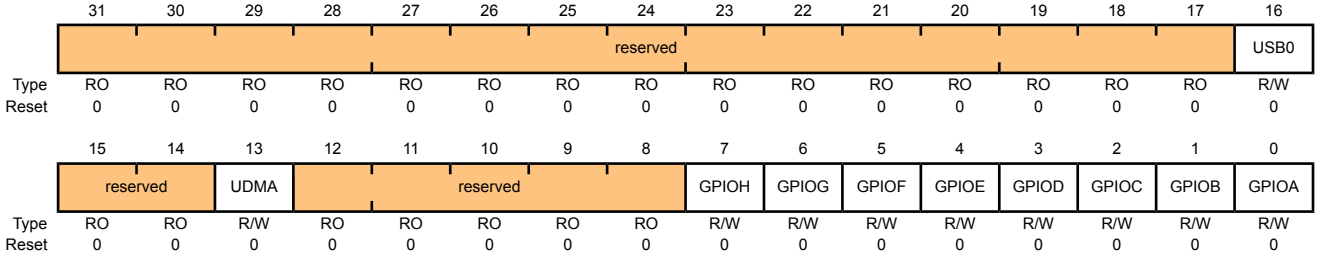
Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
25	COMP1	R/W	0	Analog Comp 1 Reset Control. Reset control for analog comparator 1.
24	COMP0	R/W	0	Analog Comp 0 Reset Control. Reset control for analog comparator 0.
23:20	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	TIMER3	R/W	0	Timer 3 Reset Control. Reset control for General-Purpose Timer module 3.
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	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14	I2C1	R/W	0	I2C1 Reset Control. Reset control for I2C unit 1.
13	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	I2C0	R/W	0	I2C0 Reset Control. Reset control for I2C unit 0.
11:9	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	QEI0	R/W	0	QEI0 Reset Control. Reset control for QEI unit 0.

Bit/Field	Name	Type	Reset	Description
7:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	SSI1	R/W	0	SSI1 Reset Control. Reset control for SSI unit 1.
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 34: 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



Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16	USB0	R/W	0	USB0 Reset Control. Reset control for USB unit 0.
15:14	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	UDMA	R/W	0	UDMA Reset Control. Reset control for uDMA unit.
12:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	GPIOH	R/W	0	Port H Reset Control. Reset control for GPIO Port H.
6	GPIOG	R/W	0	Port G Reset Control. Reset control for GPIO Port G.
5	GPIOF	R/W	0	Port F Reset Control. Reset control for GPIO Port F.
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 Hibernation Module

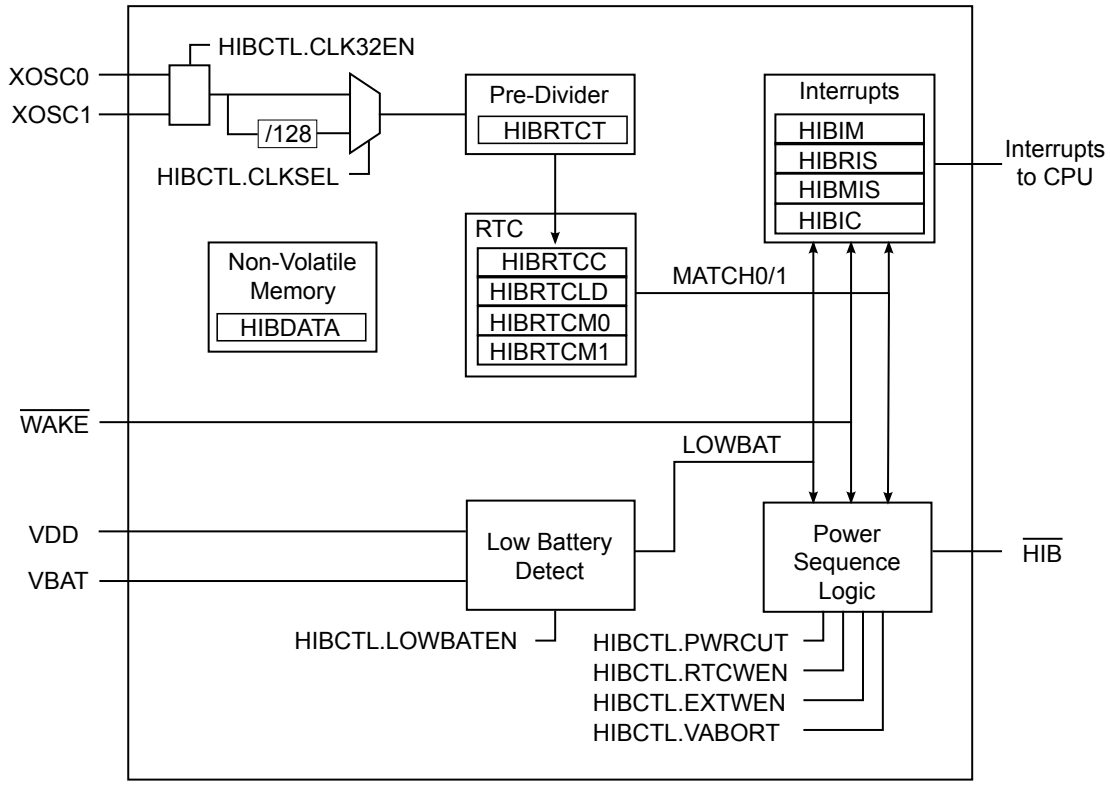
The Hibernation Module manages removal and restoration of power to provide a means for reducing power consumption. When the processor and peripherals are idle, power can be completely removed with only the Hibernation module remaining powered. Power can be restored based on an external signal, or at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

The Hibernation module has the following features:

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

7.1 Block Diagram

Figure 7-1. Hibernation Module Block Diagram



7.2 Functional Description

The Hibernation module controls the power to the processor with an enable signal (HIB) that signals an external voltage regulator to turn off.

The Hibernation module power source is determined dynamically. The supply voltage of the Hibernation module is the larger of the main voltage source (VDD) or the battery/auxilliary voltage source (VBAT). A voting circuit indicates the larger and an internal power switch selects the appropriate voltage source. The Hibernation module also has a separate clock source to maintain a real-time clock (RTC). Once in hibernation, the module signals an external voltage regulator to turn back on the power when an external pin (WAKE) is asserted, or when the internal RTC reaches a certain value. The Hibernation module can also detect when the battery voltage is low, and optionally prevent hibernation when this occurs.

Power-up from a power cut to code execution is defined as the regulator turn-on time (specified at $t_{HIB\_TO\_VDD}$ maximum) plus the normal chip POR (see “Hibernation Module” on page 756).

7.2.1 Register Access Timing

Because the Hibernation module has an independent clocking domain, certain registers must be written only with a timing gap between accesses. The delay time is $t_{HIB\_REG\_WRITE}$, therefore software must guarantee that a delay of $t_{HIB\_REG\_WRITE}$ is inserted between back-to-back writes to certain Hibernation registers, or between a write followed by a read to those same registers. There is no restriction on timing for back-to-back reads from the Hibernation module. Software may make use

of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. This bit is cleared on a write operation and set once the write completes, indicating to software that another write or read may be started safely. Software should poll **HIBCTL** for WRC=1 prior to accessing any affected register. The following registers are subject to this timing restriction:

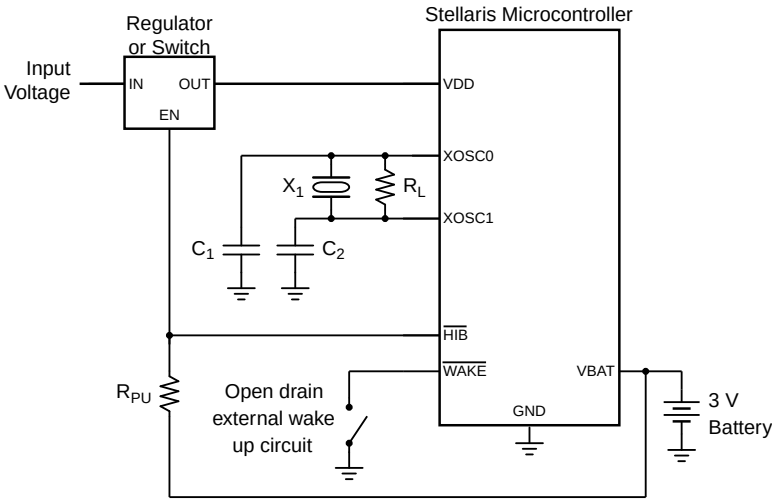
- **Hibernation RTC Counter (HIBRTCC)**
- **Hibernation RTC Match 0 (HIBRTCM0)**
- **Hibernation RTC Match 1 (HIBRTCM1)**
- **Hibernation RTC Load (HIBRTCLD)**
- **Hibernation RTC Trim (HIBRTCT)**
- **Hibernation Data (HIBDATA)**

7.2.2 Clock Source

The Hibernation module must be clocked by an external source, even if the RTC feature is not used. An external oscillator or crystal can be used for this purpose. To use a crystal, a 4.194304-MHz crystal is connected to the XOSC0 and XOSC1 pins. This clock signal is divided by 128 internally to produce the 32.768-kHz clock reference. For an alternate clock source, a 32.768-kHz oscillator can be connected to the XOSC0 pin. See Figure 7-2 on page 146 and Figure 7-3 on page 146. Note that these diagrams only show the connection to the Hibernation pins and not to the full system. See “Hibernation Module” on page 756 for specific values.

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

Figure 7-2. Clock Source Using Crystal



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

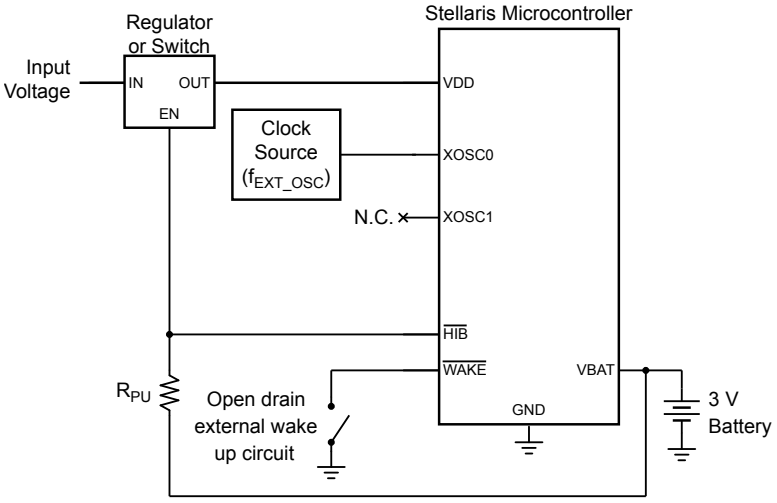
$C_{1,2}$ = Capacitor value derived from crystal vendor load capacitance specifications.

R_L = Load resistor is R_{XOSC_LOAD}.

R_{PU} = Pull-up resistor (1 M½).

See “Hibernation Module” on page 756 for specific parameter values.

Figure 7-3. Clock Source Using Dedicated Oscillator



Note: R_{PU} = Pull-up resistor (1 M½).

7.2.3 Battery Management

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

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

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

The Hibernation module can be configured to detect a low battery condition by setting the LOWBATEN bit of the **HIBCTL** register. In this configuration, the LOWBAT bit of the **HIBRIS** register will be set when the battery level is low. If the VABORT bit is also set, then the module is prevented from entering Hibernation mode when a low battery is detected. The module can also be configured to generate an interrupt for the low-battery condition (see "Interrupts and Status" on page 148).

7.2.4 Real-Time Clock

The Hibernation module includes a 32-bit counter that increments once per second with a proper clock source and configuration (see "Clock Source" on page 145). The 32.768-kHz clock signal is fed into a predivider register which counts down the 32.768-kHz clock ticks to achieve a once per second clock rate for the RTC. The rate can be adjusted to compensate for inaccuracies in the clock source by using the predivider trim register, **HIBRTCT**. This register has a nominal value of 0x7FFF, and is used for one second out of every 64 seconds to divide the input clock. This allows the software to make fine corrections to the clock rate by adjusting the predivider trim register up or down from 0x7FFF. The predivider trim should be adjusted up from 0x7FFF in order to slow down the RTC rate, and down from 0x7FFF in order to speed up the RTC rate.

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

The RTC must be enabled with the RTCEN bit of the **HIBCTL** register. The value of the RTC can be set at any time by writing to the **HIBRTCLD** register. The predivider trim can be adjusted by reading and writing the **HIBRTCT** register. The predivider uses this register once every 64 seconds to adjust the clock rate. The two match registers can be set by writing to the **HIBRTCM0** and **HIBRTCM1** registers. The RTC can be configured to generate interrupts by using the interrupt registers (see "Interrupts and Status" on page 148).

7.2.5 Non-Volatile Memory

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

7.2.6 Power Control

Important: The Hibernation Module requires special system implementation considerations when using $\overline{\text{HIB}}$ to control power, as it is intended to power-down all other sections of its host device. All system signals and power supplies that connect to the chip must be driven to 0 V_{DC} or powered down with the same regulator controlled by $\overline{\text{HIB}}$. See "Hibernation Module" on page 756 for more details.

The Hibernation module controls power to the microcontroller through the use of the $\overline{\text{HIB}}$ pin. This pin is intended to be connected to the enable signal of the external regulator(s) providing 3.3 V

and/or 2.5 V to the microcontroller. When the HIB signal is asserted by the Hibernation module, the external regulator is turned off and no longer powers the system. The Hibernation module remains powered from the VBAT supply (which could be a battery or an auxiliary power source) until a Wake event. Power to the device is restored by deasserting the HIB signal, which causes the external regulator to turn power back on to the chip.

7.2.7 Initiating Hibernate

Hibernation mode is initiated by the microcontroller setting the HIBREQ bit of the **HIBCTL** register. Prior to doing this, a wake-up condition must be configured, either from the external WAKE pin, or by using an RTC match.

The Hibernation module is configured to wake from the external WAKE pin by setting the PINWEN bit of the **HIBCTL** register. It is configured to wake from RTC match by setting the RTCWEN bit. Either one or both of these bits can be set prior to going into hibernation. The WAKE pin includes a weak internal pull-up. Note that both the HIB and WAKE pins use the Hibernation module's internal power supply as the logic 1 reference.

When the Hibernation module wakes, the microcontroller will see a normal power-on reset. Software can detect that the power-on was due to a wake from hibernation by examining the raw interrupt status register (see “Interrupts and Status” on page 148) and by looking for state data in the non-volatile memory (see “Non-Volatile Memory” on page 147).

When the HIB signal deasserts, enabling the external regulator, the external regulator must reach the operating voltage within $t_{HIB\_TO\_VDD}$.

7.2.8 Interrupts and Status

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

- Assertion of WAKE pin
- RTC match
- Low battery detected

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

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

7.3 Initialization and Configuration

The Hibernation module can be set in several different configurations. The following sections show the recommended programming sequence for various scenarios. The examples below assume that a 32.768-kHz oscillator is used, and thus always show bit 2 (CLKSEL) of the **HIBCTL** register set to 1. If a 4.194304-MHz crystal is used instead, then the CLKSEL bit remains cleared. Because the Hibernation module runs at 32.768 kHz and is asynchronous to the rest of the system, software must allow a delay of $t_{HIB\_REG\_WRITE}$ after writes to certain registers (see “Register Access Timing” on page 144). The registers that require a delay are listed in a note in “Register Map” on page 150 as well as in each register description.

7.3.1 Initialization

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

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

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

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

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

7.3.2 RTC Match Functionality (No Hibernation)

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

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

7.3.3 RTC Match/Wake-Up from Hibernation

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

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

7.3.4 External Wake-Up from Hibernation

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

1. Write any data to be retained during power cut to the **HIBDATA** register at offsets 0x030-0x12C.

2. Enable the external wake and start the hibernation sequence by writing 0x0000.0056 to the **HIBCTL** register at offset 0x010.

7.3.5 RTC/External Wake-Up from Hibernation

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

7.3.6 Register Reset

The Hibernation module handles resets according to the following conditions:

- Cold Reset

  When the hibernation module has no externally applied voltage and detects a change to either VDD or VBAT, it resets all hibernation module registers to the value in Table 7-1 on page 151.

- Reset During Hibernation Module Disable

  When the module has either not been enabled or has been disabled by software, the reset is passed through to the Hibernation module circuitry, and the internal state of the module is reset.

- Reset While Hibernation Module is in Hibernation Mode

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

- Reset While Hibernation Module is in Normal Mode

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

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

7.4 Register Map

Table 7-1 on page 151 lists the Hibernation registers. All addresses given are relative to the Hibernation Module base address at 0x400F.C000.

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 144.

Table 7-1. Hibernation Module Register Map

Offset	Name	Type	Reset	Description	See page
0x000	HIBRTCC	RO	0x0000.0000	Hibernation RTC Counter	152
0x004	HIBRTCM0	R/W	0xFFFF.FFFF	Hibernation RTC Match 0	153
0x008	HIBRTCM1	R/W	0xFFFF.FFFF	Hibernation RTC Match 1	154
0x00C	HIBRTCLD	R/W	0xFFFF.FFFF	Hibernation RTC Load	155
0x010	HIBCTL	R/W	0x0000.0000	Hibernation Control	156
0x014	HIBIM	R/W	0x0000.0000	Hibernation Interrupt Mask	159
0x018	HIBRIS	RO	0x0000.0000	Hibernation Raw Interrupt Status	160
0x01C	HIBMIS	RO	0x0000.0000	Hibernation Masked Interrupt Status	161
0x020	HIBIC	R/W1C	0x0000.0000	Hibernation Interrupt Clear	162
0x024	HIBRTCT	R/W	0x0000.7FFF	Hibernation RTC Trim	163
0x030-0x12C	HIBDATA	R/W	0x0000.0000	Hibernation Data	164

7.5 Register Descriptions

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

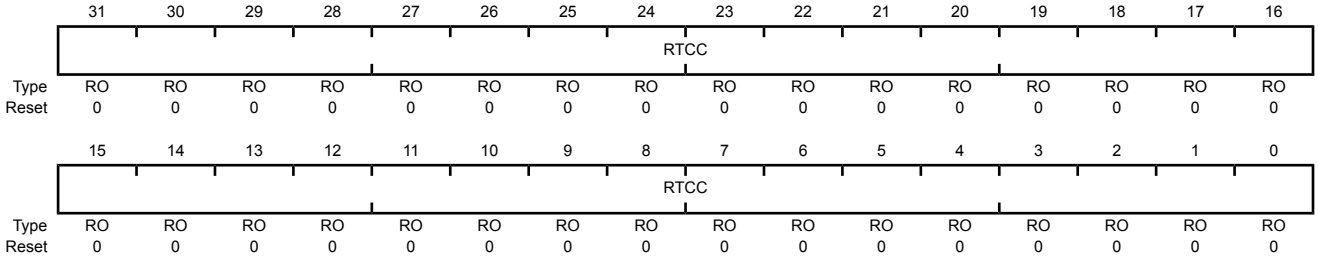
Register 1: Hibernation RTC Counter (HIBRTCC), offset 0x000

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

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 144.

Hibernation RTC Counter (HIBRTCC)

Base 0x400F.C000
Offset 0x000
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:0	RTCC	RO	0x0000.0000	RTC Counter A read returns the 32-bit counter value. This register is read-only. To change the value, use the **HIBRTCLD** register.

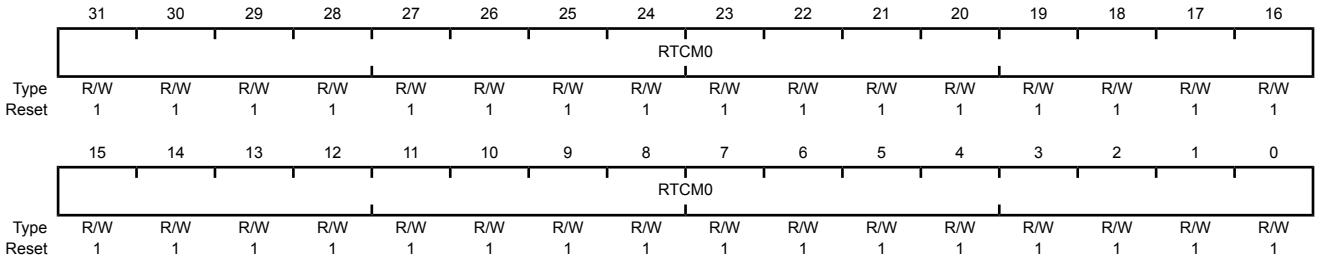
Register 2: Hibernation RTC Match 0 (HIBRTCM0), offset 0x004

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

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 144.

Hibernation RTC Match 0 (HIBRTCM0)

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



Bit/Field	Name	Type	Reset	Description
31:0	RTCM0	R/W	0xFFFF.FFFF	RTC Match 0 A write loads the value into the RTC match register. A read returns the current match value.

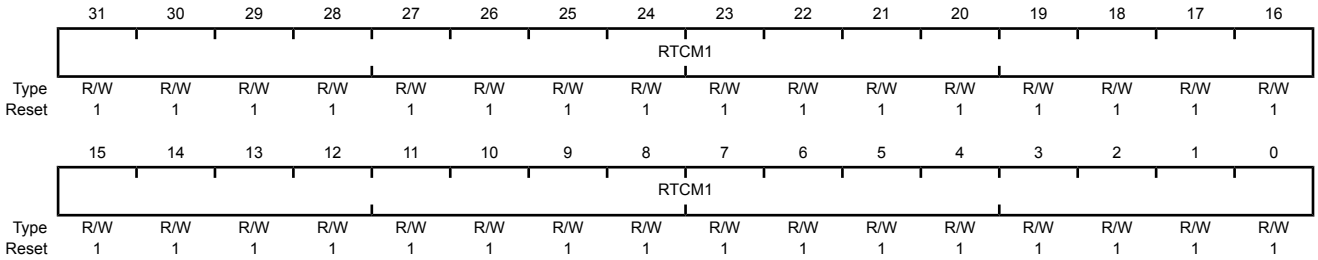
Register 3: Hibernation RTC Match 1 (HIBRTCM1), offset 0x008

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

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 144.

Hibernation RTC Match 1 (HIBRTCM1)

Base 0x400F.C000
Offset 0x008
Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	RTCM1	R/W	0xFFFF.FFFF	RTC Match 1 A write loads the value into the RTC match register. A read returns the current match value.

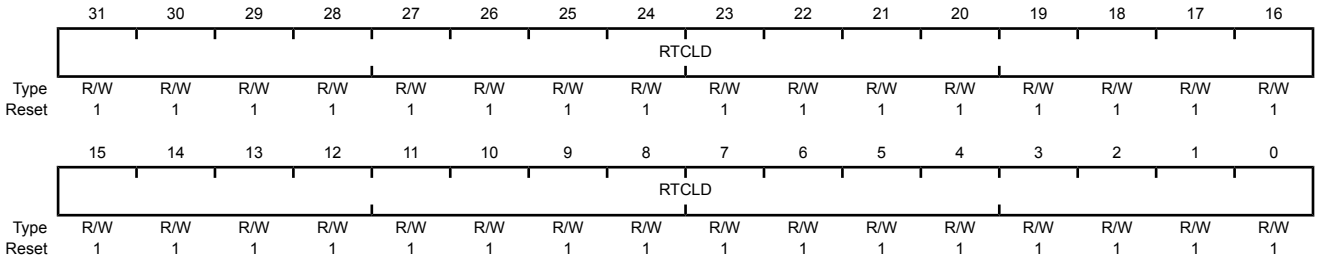
Register 4: Hibernation RTC Load (HIBRTCLD), offset 0x00C

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

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 144.

Hibernation RTC Load (HIBRTCLD)

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



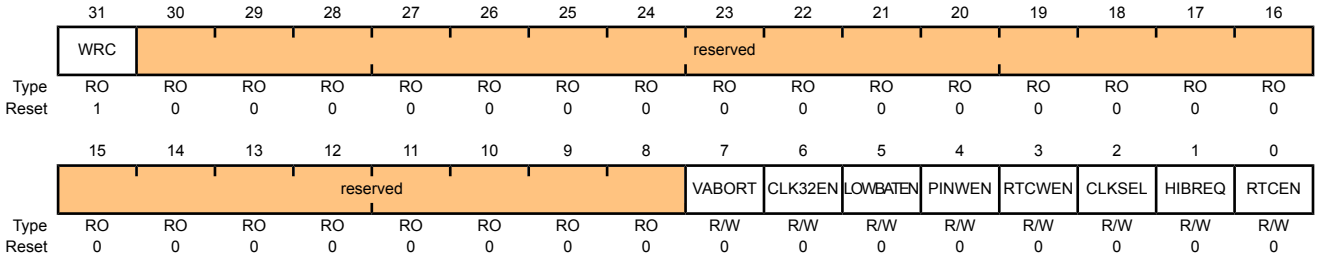
Bit/Field	Name	Type	Reset	Description
31:0	RTCLD	R/W	0xFFFF.FFFF	RTC Load A write loads the current value into the RTC counter (RTCC). A read returns the 32-bit load value.

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

This register is the control register for the Hibernation module.

Hibernation Control (HIBCTL)

Base 0x400F.C000
Offset 0x010
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31	WRC	RO	1	Write Complete/Capable This bit indicates whether the hibernation module can receive a write operation. Value Description 0 The interface is processing a prior write and is busy. Any write operation that is attempted while WRC is 0 results in undetermined behavior. 1 The interface is ready to accept a write. Software must poll this bit between write requests and defer writes until WRC=1 to ensure proper operation. This difference may be exploited by software at reset time to detect which method of programming is appropriate: 0 = software delay loops required; 1 = WRC paced available. The bit name WRC means "Write Complete," which is the normal use of the bit (between write accesses). However, because the bit is set out-of-reset, the name can also mean "Write Capable" which simply indicates that the interface may be written to by software. This meaning also has more meaning to the out-of-reset sense.
30:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	VABORT	R/W	0	Power Cut Abort Enable Value Description 0 Power cut occurs during a low-battery alert. 1 Power cut is aborted.

Bit/Field	Name	Type	Reset	Description
6	CLK32EN	R/W	0	Clocking Enable Value Description 0 Disabled 1 Enabled This bit must be enabled to use the Hibernation module. If a crystal is used, then software should wait 20 ms after setting this bit to allow the crystal to power up and stabilize.
5	LOWBATEN	R/W	0	Low Battery Monitoring Enable Value Description 0 Disabled 1 Enabled When set, low battery voltage detection is enabled (VBAT < V_{LOWBAT}).
4	PINWEN	R/W	0	External $\overline{\text{WAKE}}$ Pin Enable Value Description 0 Disabled 1 Enabled When set, an external event on the $\overline{\text{WAKE}}$ pin will re-power the device.
3	RTCWEN	R/W	0	RTC Wake-up Enable Value Description 0 Disabled 1 Enabled When set, an RTC match event (RTCM0 or RTCM1) will re-power the device based on the RTC counter value matching the corresponding match register 0 or 1.
2	CLKSEL	R/W	0	Hibernation Module Clock Select Value Description 0 Use Divide by 128 output. Use this value for a 4.194304-MHz crystal. 1 Use raw output. Use this value for a 32.768-kHz oscillator.
1	HIBREQ	R/W	0	Hibernation Request Value Description 0 Disabled 1 Hibernation initiated After a wake-up event, this bit is cleared by hardware.

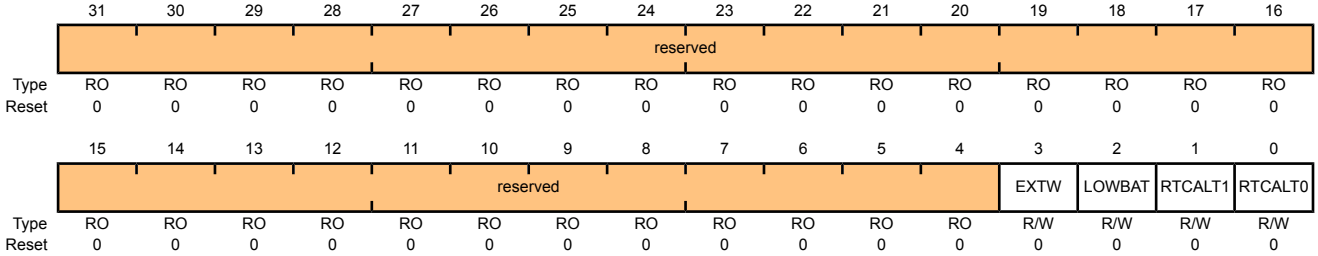
Bit/Field	Name	Type	Reset	Description
0	RTCEN	R/W	0	RTC Timer Enable Value Description 0 Disabled 1 Enabled

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

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

Hibernation Interrupt Mask (HIBIM)

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



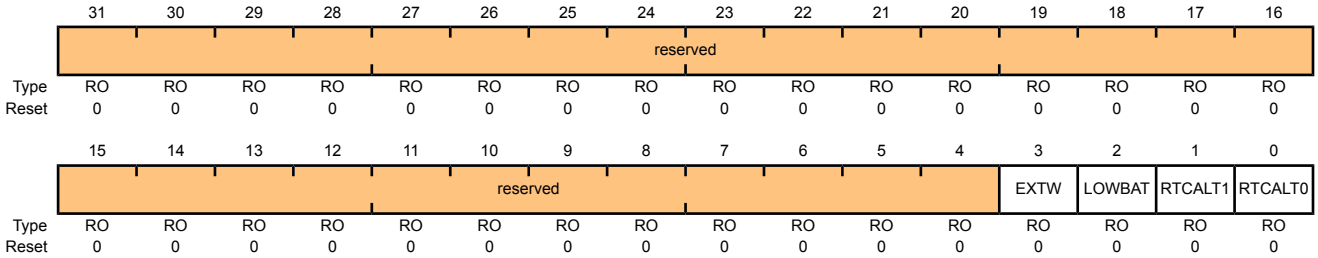
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	R/W	0	External Wake-Up Interrupt Mask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

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

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

Hibernation Raw Interrupt Status (HIBRIS)

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



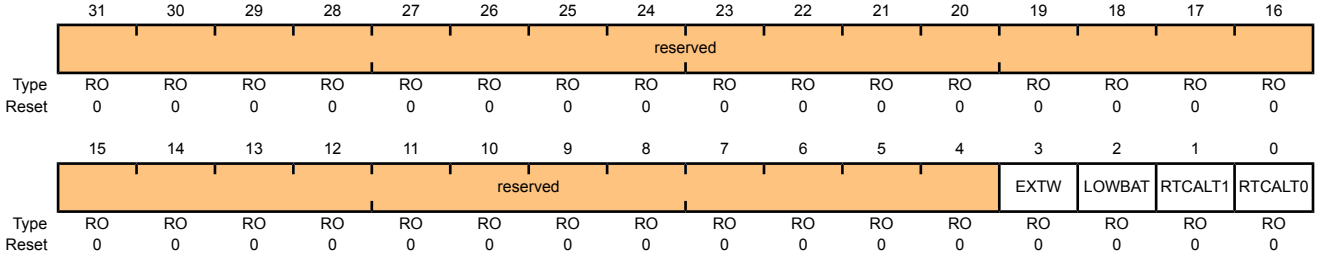
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	RO	0	External Wake-Up Raw Interrupt Status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

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

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

Hibernation Masked Interrupt Status (HIBMIS)

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



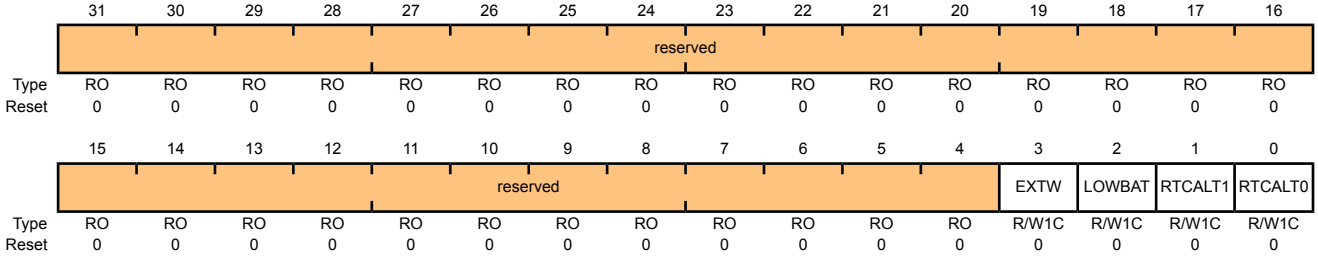
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000.0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EXTW	RO	0	External Wake-Up Masked Interrupt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

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

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

Hibernation Interrupt Clear (HIBIC)

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



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

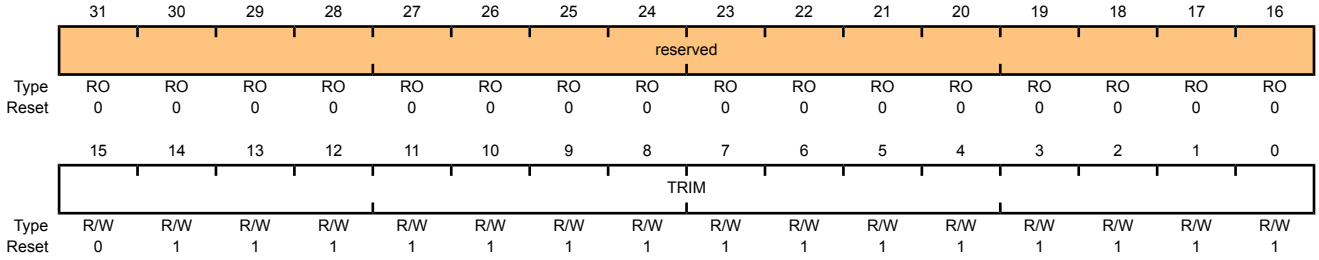
Register 10: Hibernation RTC Trim (HIBRTCT), offset 0x024

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

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 144.

Hibernation RTC Trim (HIBRTCT)

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



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

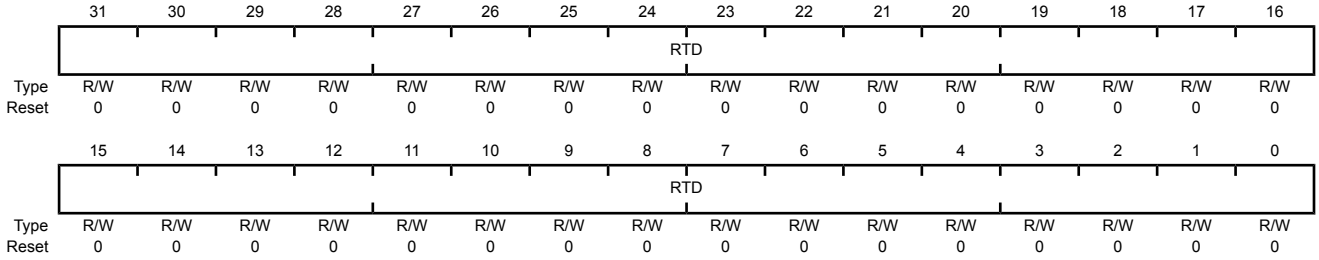
Register 11: Hibernation Data (HIBDATA), offset 0x030-0x12C

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

Note: **HIBRTCC**, **HIBRTCM0**, **HIBRTCM1**, **HIBRTCLD**, **HIBRTCT**, and **HIBDATA** are on the Hibernation module clock domain and and have special timing requirements. Software should make use of the WRC bit in the **HIBCTL** register to ensure that the required timing gap has elapsed. See “Register Access Timing” on page 144.

Hibernation Data (HIBDATA)

Base 0x400F.C000
Offset 0x030-0x12C
Type R/W, reset 0x0000.0000



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

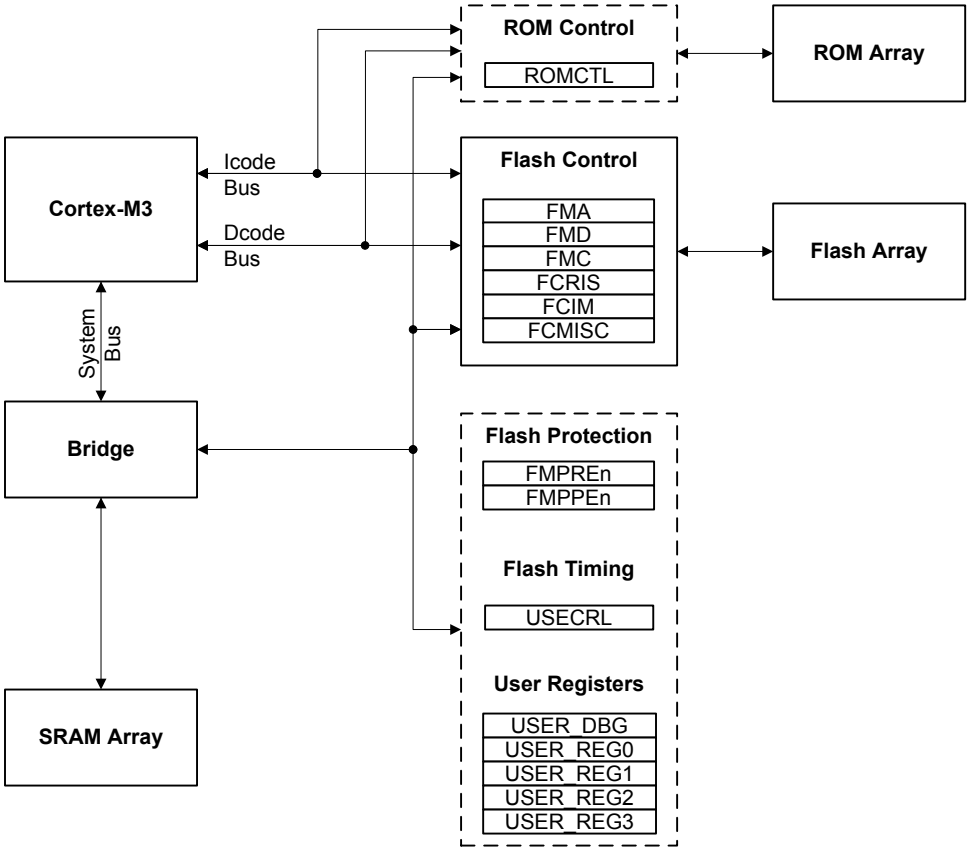
8 Internal Memory

The LM3S5749 microcontroller comes with 64 KB of bit-banded SRAM and 128 KB of flash memory. The flash controller provides a user-friendly interface, making flash programming a simple task. Flash protection can be applied to the flash memory on a 2-KB block basis.

8.1 Block Diagram

Figure 8-1 on page 165 illustrates the Flash functions. The dashed boxes in the figure indicate registers residing in the System Control module rather than the Flash Control module.

Figure 8-1. Flash Block Diagram



8.2 Functional Description

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

8.2.1 SRAM Memory

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

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

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

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

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

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

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

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

8.2.2 ROM Memory

The 11 KB of internal ROM of the Stellaris® device is located at address 0x0100.0000 of the device memory map and contains the following components:

- Stellaris® Boot Loader and vector table (see “Boot Loader” on page 764)
- Stellaris® Peripheral Driver Library (DriverLib) release for product-specific peripherals and interfaces (see “ROM DriverLib Functions” on page 769)

8.2.3 Flash Memory

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

8.2.3.1 Flash Memory Timing

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

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

8.2.3.2 Flash Memory Protection

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

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

The policies may be combined as shown in Table 8-1 on page 167.

Table 8-1. Flash Protection Policy Combinations

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

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

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

The factory settings for the **FMPREn** and **FMPPEn** registers are a value of 1 for all implemented banks. This implements a policy of open access and programmability. The register bits may be changed by writing the specific register bit. The changes are not permanent until the register is committed (saved), at which point the bit change is permanent. If a bit is changed from a 1 to a 0 and not committed, it may be restored by executing a power-on reset sequence. Details on programming these bits are discussed in “Nonvolatile Register Programming” on page 168.

8.3 Flash Memory Initialization and Configuration

8.3.1 Flash Programming

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

8.3.1.1 To program a 32-bit word

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

4. Poll the **FMC** register until the WRITE bit is cleared.

8.3.1.2 To perform an erase of a 1-KB page

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

8.3.1.3 To perform a mass erase of the flash

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

8.3.2 Nonvolatile Register Programming

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

Important: These registers can only have bits changed from 1 to 0 by user programming, but can be restored to their factory default values by performing the sequence described in the section called “Recovering a "Locked" Device” on page 65. The mass erase of the main flash array caused by the sequence is performed prior to restoring these registers.

In addition, the **USER_REG0**, **USER_REG1**, and **USER_DBG** use bit 31 (NW) of their respective registers to indicate that they are available for user write. These three registers can only be written once whereas the flash protection registers may be written multiple times. Table 8-2 on page 168 provides the FMA address required for commitment of each of the registers and the source of the data to be written when the COMT bit of the **FMC** register is written with a value of 0xA442.0008. After writing the COMT bit, the user may poll the **FMC** register to wait for the commit operation to complete.

Table 8-2. Flash Resident Registers[a]

Register to be Committed	FMA Value	Data Source
FMPRE0	0x0000.0000	FMPRE0
FMPRE1	0x0000.0002	FMPRE1
FMPRE2	0x0000.0004	FMPRE2
FMPRE3	0x0000.0006	FMPRE3
FMPPE0	0x0000.0001	FMPPE0
FMPPE1	0x0000.0003	FMPPE1
FMPPE2	0x0000.0005	FMPPE2
FMPPE3	0x0000.0007	FMPPE3
USER_REG0	0x8000.0000	USER_REG0
USER_REG1	0x8000.0001	USER_REG1

Register to be Committed	FMA Value	Data Source
USER_DBG	0x7510.0000	FMD

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

8.4 Register Map

Table 8-3 on page 169 lists the ROM Controller registers and the Flash memory and control registers. The offset listed is a hexadecimal increment to the register's address. The ROM Controller registers are relative to the System Control base address of 0x400F.E000. 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 8-3. Flash Register Map

Offset	Name	Type	Reset	Description	See page
ROM Registers (System Control Offset)					
0x0F0	RMCTL	R/W1C	-	ROM Control	171
Flash Registers (Flash Control Offset)					
0x000	FMA	R/W	0x0000.0000	Flash Memory Address	172
0x004	FMD	R/W	0x0000.0000	Flash Memory Data	173
0x008	FMC	R/W	0x0000.0000	Flash Memory Control	174
0x00C	FCRIS	RO	0x0000.0000	Flash Controller Raw Interrupt Status	176
0x010	FCIM	R/W	0x0000.0000	Flash Controller Interrupt Mask	177
0x014	FCMISC	R/W1C	0x0000.0000	Flash Controller Masked Interrupt Status and Clear	178
Flash Registers (System Control Offset)					
0x0F4	RMVER	RO	0x0000.0000	ROM Version Register	180
0x130	FMPRE0	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 0	181
0x200	FMPRE0	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 0	181
0x134	FMPPE0	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 0	182
0x400	FMPPE0	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 0	182
0x140	USECRL	R/W	0x31	USec Reload	179
0x1D0	USER_DBG	R/W	0xFFFF.FFFE	User Debug	183
0x1E0	USER_REG0	R/W	0xFFFF.FFFF	User Register 0	184
0x1E4	USER_REG1	R/W	0xFFFF.FFFF	User Register 1	185
0x1E8	USER_REG2	R/W	0xFFFF.FFFF	User Register 2	186
0x1EC	USER_REG3	R/W	0xFFFF.FFFF	User Register 3	187
0x204	FMPRE1	R/W	0xFFFF.FFFF	Flash Memory Protection Read Enable 1	188
0x208	FMPRE2	R/W	0x0000.0000	Flash Memory Protection Read Enable 2	189

Offset	Name	Type	Reset	Description	See page
0x20C	FMPRE3	R/W	0x0000.0000	Flash Memory Protection Read Enable 3	190
0x404	FMPPE1	R/W	0xFFFF.FFFF	Flash Memory Protection Program Enable 1	191
0x408	FMPPE2	R/W	0x0000.0000	Flash Memory Protection Program Enable 2	192
0x40C	FMPPE3	R/W	0x0000.0000	Flash Memory Protection Program Enable 3	193

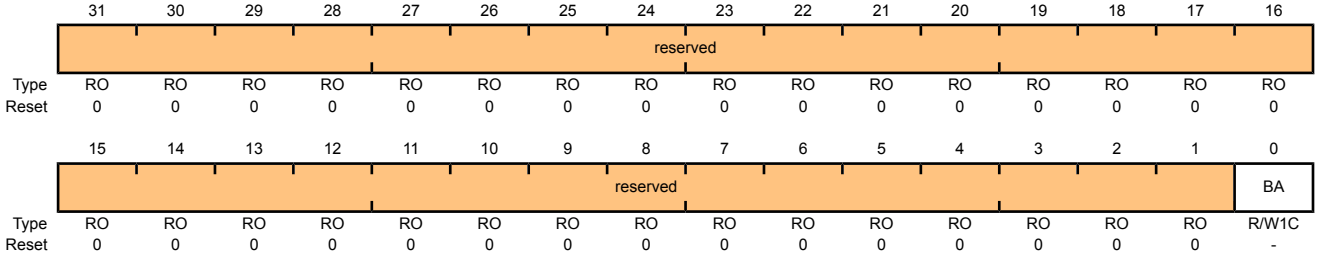
8.5 ROM Register Descriptions (System Control Offset)

This section lists and describes the ROM Controller registers, in numerical order by address offset. Registers in this section are relative to the System Control base address of 0x400F.E000.

Register 1: ROM Control (RMCTL), offset 0x0F0

This register provides control of the ROM controller state.

ROM Control (RMCTL)
Base 0x400F.E000
Offset 0x0F0
Type R/W1C, reset -



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	BA	R/W1C	-	Boot Alias ■ The device has ROM. ■ The first two words of the Flash memory contain 0xFFFF.FFFF. This bit is cleared by writing a 1 to this bit position. When the BA bit is set, the boot alias is in effect and the ROM appears at address 0x0. When the BA bit is clear, the Flash appears at address 0x0.

8.6 Flash Register Descriptions (Flash Control Offset)

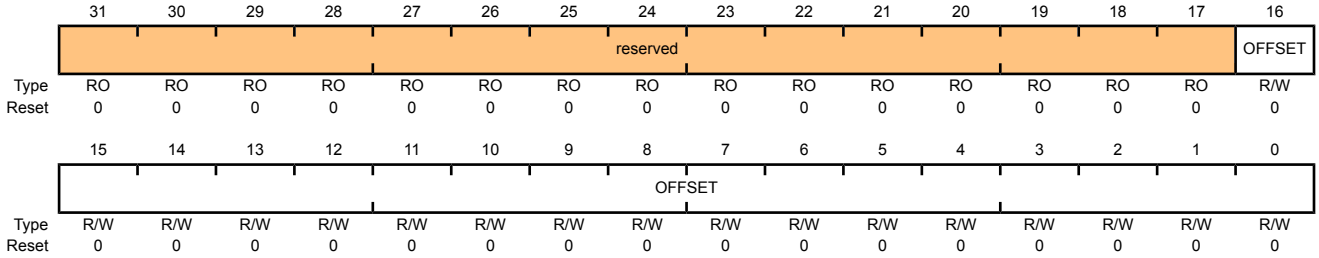
This section lists and describes the Flash Memory registers, in numerical order by address offset. Registers in this section are relative to the Flash control base address of 0x400F.D000.

Register 2: Flash Memory Address (FMA), offset 0x000

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

Flash Memory Address (FMA)

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



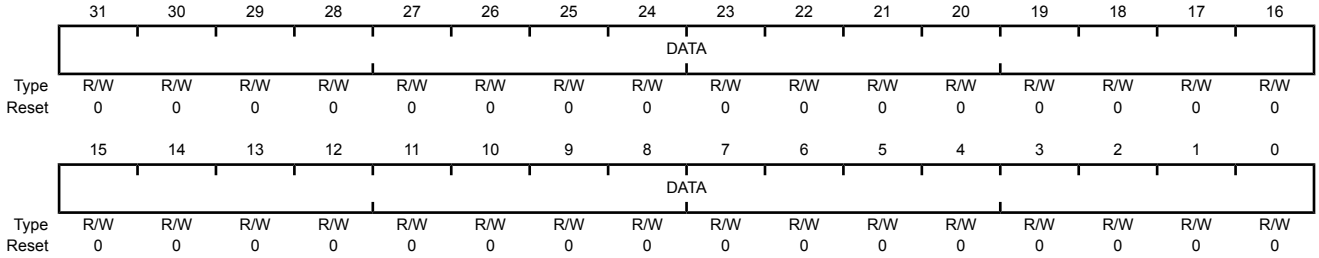
Bit/Field	Name	Type	Reset	Description
31:17	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
16:0	OFFSET	R/W	0x0	Address Offset Address offset in flash where operation is performed, except for nonvolatile registers (see “Nonvolatile Register Programming” on page 168 for details on values for this field).

Register 3: 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



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

Register 4: Flash Memory Control (FMC), offset 0x008

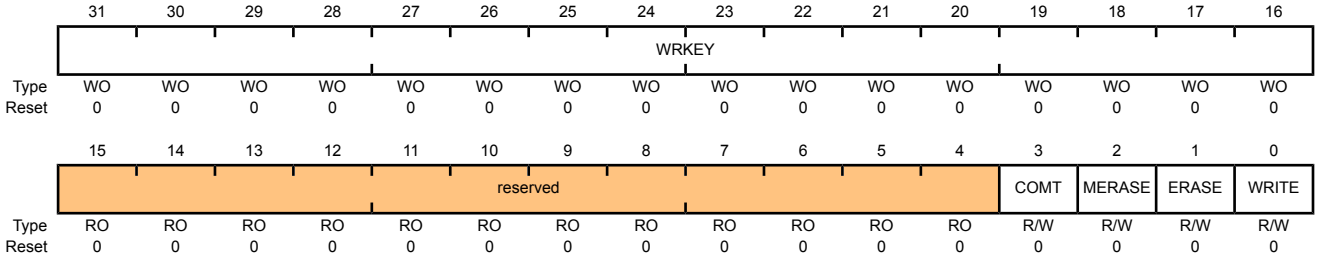
When this register is written, the flash controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 172). If the access is a write access, the data contained in the **Flash Memory Data (FMD)** register (see page 173) is written.

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



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.

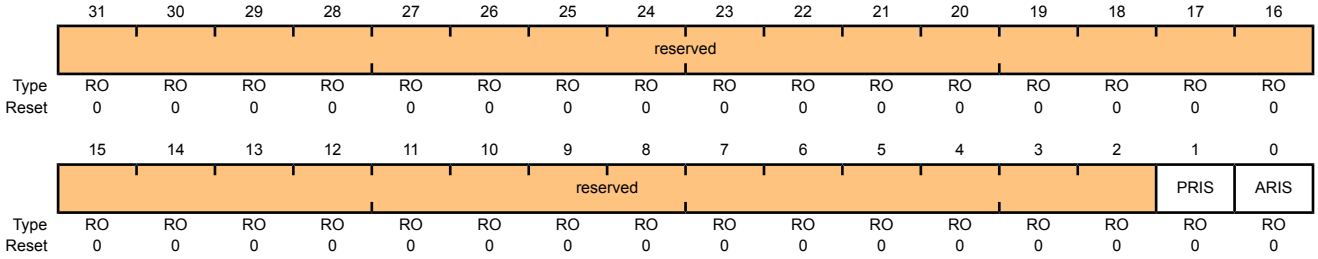
Bit/Field	Name	Type	Reset	Description
1	ERASE	R/W	0	Erase a Page of Flash Memory If this bit is set, the page of flash main memory as specified by the contents of **FMA** is erased. A write of 0 has no effect on the state of this bit. If read, the state of the previous erase access is provided. If the 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 5: 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



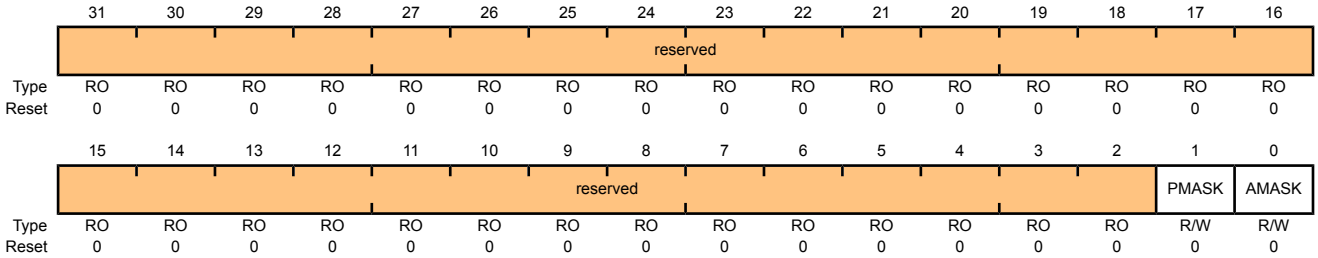
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PRIS	RO	0	Programming Raw Interrupt Status This bit 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 174).
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 6: 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



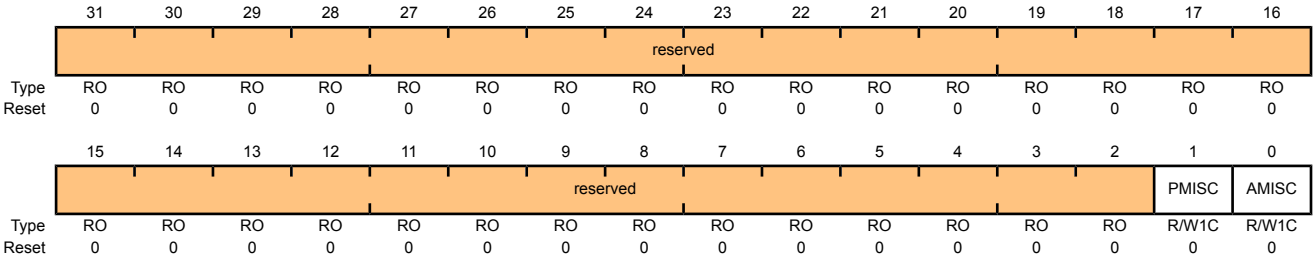
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	PMASK	R/W	0	Programming Interrupt Mask This bit controls 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 7: 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



Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	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 176) 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.

8.7 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.

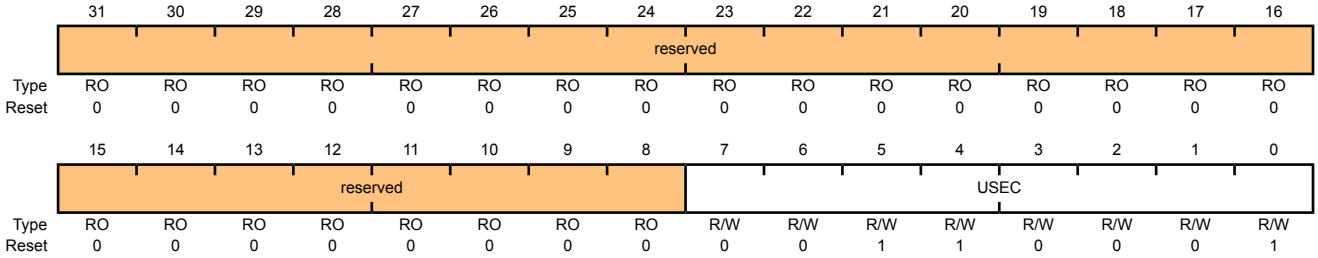
Register 8: USec Reload (USECRL), offset 0x140

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



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	USEC	R/W	0x31	Microsecond Reload Value MHz -1 of the controller clock when the flash is being erased or programmed. If the maximum system frequency is being used, USEC should be set to 0x31 (50 MHz) whenever the flash is being erased or programmed.

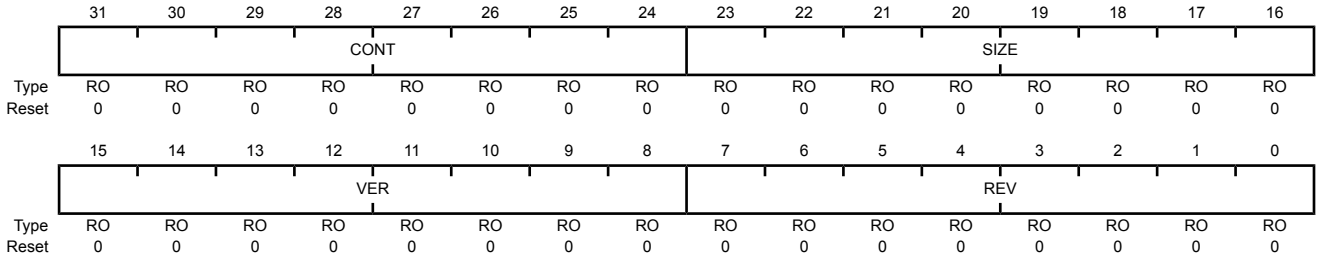
Register 9: ROM Version Register (RMVER), offset 0x0F4

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

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

ROM Version Register (RMVER)

Base 0x400F.E000
Offset 0x0F4
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:24	CONT	RO	0x0	ROM Contents This field specifies the contents of the ROM. Value Description 0x0 Stellaris Boot Loader & DriverLib
23:16	SIZE	RO	0x0	ROM Size This field encodes the size of the ROM. Value Description 0x0 11 KB
15:8	VER	RO	0x0	ROM Version
7:0	REV	RO	0x0	ROM Revision

Register 10: Flash Memory Protection Read Enable 0 (FMPRE0), offset 0x130 and 0x200

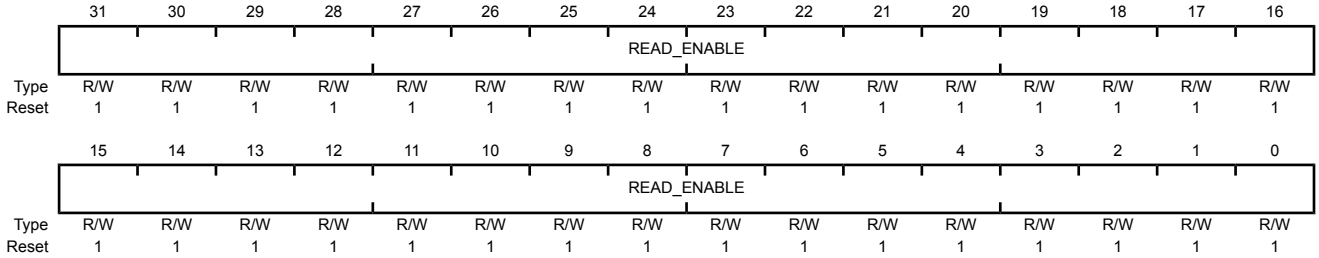
Note: This register is aliased for backwards compatability.

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

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

Flash Memory Protection Read Enable 0 (FMPRE0)

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



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

Value	Description
0xFFFFFFFF	Enables 128 KB of flash.

Register 11: Flash Memory Protection Program Enable 0 (FMPPE0), offset 0x134 and 0x400

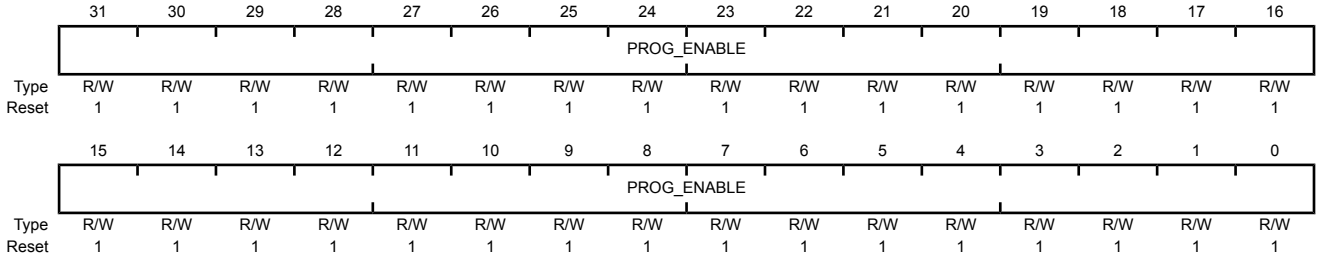
Note: This register is aliased for backwards compatability.

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

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

Flash Memory Protection Program Enable 0 (FMPPE0)

Base 0x400F.E000
Offset 0x134 and 0x400
Type R/W, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash Programming Enable Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”. Value Description 0xFFFFFFFF Enables 128 KB of flash.

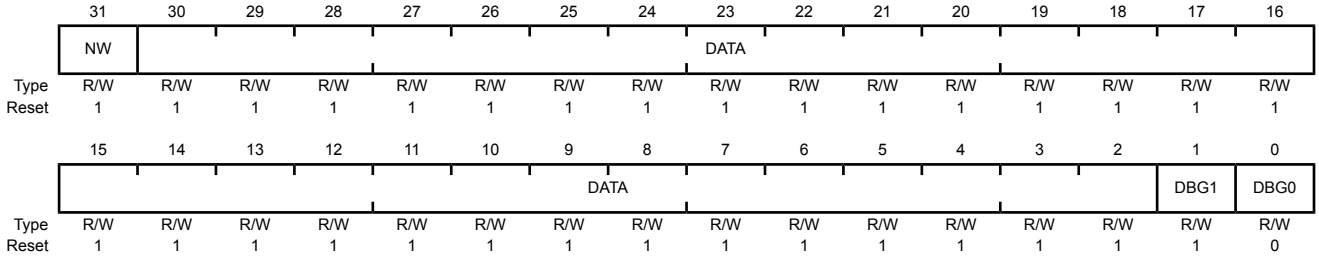
Register 12: User Debug (USER_DBG), offset 0x1D0

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

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

User Debug (USER_DBG)

Base 0x400F.E000
Offset 0x1D0
Type R/W, reset 0xFFFF.FFFE



Bit/Field	Name	Type	Reset	Description
31	NW	R/W	1	User Debug Not Written. Specifies that this 32-bit dword has not been written.
30:2	DATA	R/W	0x1FFFFFFF	User Data. Contains the user data value. This field is initialized to all 1s and can only be written once.
1	DBG1	R/W	1	Debug Control 1. The DBG1 bit must be 1 and DBG0 must be 0 for debug to be available.
0	DBG0	R/W	0	Debug Control 0. The DBG1 bit must be 1 and DBG0 must be 0 for debug to be available.

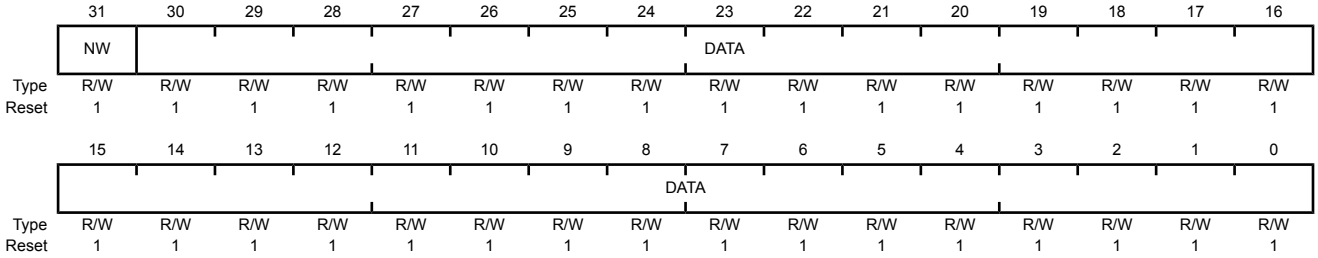
Register 13: User Register 0 (USER_REG0), offset 0x1E0

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

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

User Register 0 (USER_REG0)

Base 0x400F.E000
Offset 0x1E0
Type R/W, reset 0xFFFF.FFFF



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

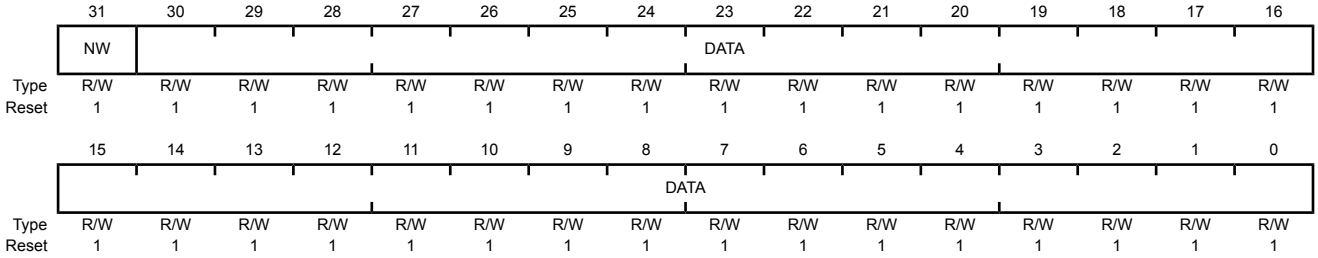
Register 14: User Register 1 (USER_REG1), offset 0x1E4

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

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

User Register 1 (USER_REG1)

Base 0x400F.E000
Offset 0x1E4
Type R/W, reset 0xFFFF.FFFF



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

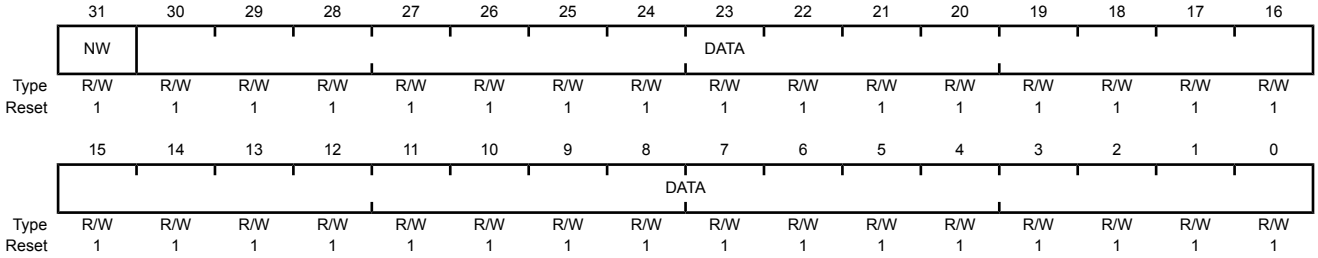
Register 15: User Register 2 (USER_REG2), offset 0x1E8

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

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

User Register 2 (USER_REG2)

Base 0x400F.E000
Offset 0x1E8
Type R/W, reset 0xFFFF.FFFF



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

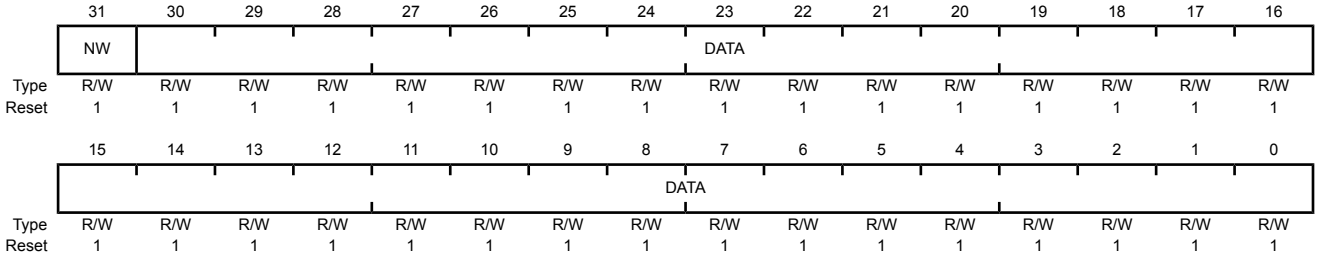
Register 16: User Register 3 (USER_REG3), offset 0x1EC

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

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

User Register 3 (USER_REG3)

Base 0x400F.E000
Offset 0x1EC
Type R/W, reset 0xFFFF.FFFF



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

Register 17: Flash Memory Protection Read Enable 1 (FMPRE1), offset 0x204

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

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

Flash Memory Protection Read Enable 1 (FMPRE1)

Base 0x400F.E000
Offset 0x204
Type R/W, reset 0xFFFF.FFFF

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

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

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

Value	Description
0xFFFFFFFF	Enables 128 KB of flash.

Register 18: Flash Memory Protection Read Enable 2 (FMPRE2), offset 0x208

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

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

Flash Memory Protection Read Enable 2 (FMPRE2)

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

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

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

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

Register 19: Flash Memory Protection Read Enable 3 (FMPRE3), offset 0x20C

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

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

Flash Memory Protection Read Enable 3 (FMPRE3)

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

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

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

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

Value	Description
0x00000000	Enables 128 KB of flash.

Register 20: Flash Memory Protection Program Enable 1 (FMPPE1), offset 0x404

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

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

Flash Memory Protection Program Enable 1 (FMPPE1)

Base 0x400F.E000
Offset 0x404
Type R/W, reset 0xFFFF.FFFF

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

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

Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0xFFFFFFFF	Flash Programming Enable. Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”. Value Description 0xFFFFFFFF Enables 128 KB of flash.

Register 21: Flash Memory Protection Program Enable 2 (FMPPE2), offset 0x408

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

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

Flash Memory Protection Program Enable 2 (FMPPE2)

Base 0x400F.E000
Offset 0x408
Type R/W, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0x00000000	Flash Programming Enable. Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”. Value Description 0x00000000 Enables 128 KB of flash.

Register 22: Flash Memory Protection Program Enable 3 (FMPPE3), offset 0x40C

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

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

Flash Memory Protection Program Enable 3 (FMPPE3)

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

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

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

Bit/Field	Name	Type	Reset	Description
31:0	PROG_ENABLE	R/W	0x00000000	Flash Programming Enable. Configures 2-KB flash blocks to be execute only. The policies may be combined as shown in the table “Flash Protection Policy Combinations”. Value Description 0x00000000 Enables 128 KB of flash.

9 Micro Direct Memory Access (µDMA)

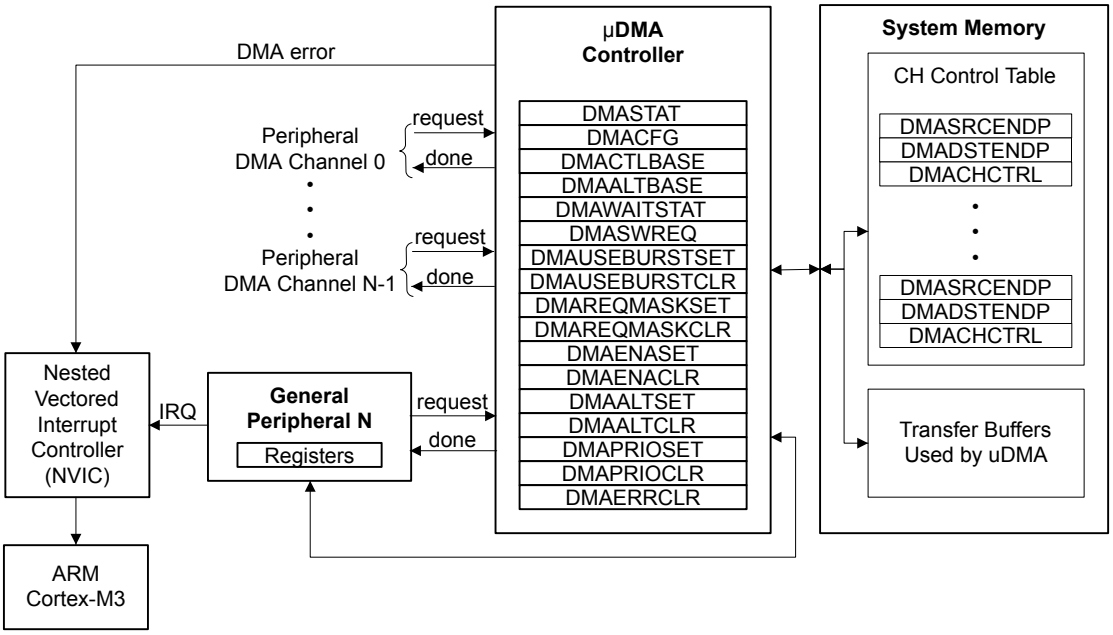
The LM3S5749 microcontroller includes a Direct Memory Access (DMA) controller, known as micro-DMA (µDMA). The µDMA controller provides a way to offload data transfer tasks from the Cortex-M3 processor, allowing for more efficient use of the processor and the expanded available bus bandwidth. The µDMA controller can perform transfers between memory and peripherals. It has dedicated channels for each supported peripheral and can be programmed to automatically perform transfers between peripherals and memory as the peripheral is ready to transfer more data. The µDMA controller also supports sophisticated transfer modes such as ping-pong and scatter-gather, which allows the processor to set up a list of transfer tasks for the controller.

The µDMA controller has the following features:

- ARM PrimeCell® 32-channel configurable µDMA controller
- Support for multiple transfer modes
  - Basic, for simple transfer scenarios
  - Ping-pong, for continuous data flow to/from peripherals
  - Scatter-gather, from a programmable list of arbitrary transfers initiated from a single request
- Dedicated channels for supported peripherals
- One channel each for receive and transmit path for bidirectional peripherals
- Dedicated channel for software-initiated transfers
- Independently configured and operated channels
- Per-channel configurable bus arbitration scheme
- Two levels of priority
- Design optimizations for improved bus access performance between µDMA controller and the processor core
  - µDMA controller access is subordinate to core access
  - RAM striping
  - Peripheral bus segmentation
- Data sizes of 8, 16, and 32 bits
- Source and destination address increment size of byte, half-word, word, or no increment
- Maskable device requests
- Optional software initiated requests for any channel
- Interrupt on transfer completion, with a separate interrupt per channel

9.1 Block Diagram

Figure 9-1. µDMA Block Diagram



9.2 Functional Description

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

Each peripheral function that is supported has a dedicated channel on the µDMA controller that can be configured independently.

The µDMA controller makes use of a unique configuration method by using channel control structures that are maintained in system memory by the processor. While simple transfer modes are supported, it is also possible to build up sophisticated "task" lists in memory that allow the controller to perform arbitrary-sized transfers to and from arbitrary locations as part of a single transfer request. The controller also supports the use of ping-pong buffering to accomodate constant streaming of data to or from a peripheral.

Each channel also has a configurable arbitration size. The arbitration size is the number of items that will be transferred in a burst before the controller rearbitrates for channel priority. Using the arbitration size, it is possible to control exactly how many items are transferred to or from a peripheral each time it makes a DMA service request.

9.2.1 Channel Assigments

µDMA channels 0-31 are assigned to peripherals according to the following table.

Note: Channels that are not listed in the table may be assigned to peripherals in the future. However, they are currently available for software use.

Table 9-1. DMA Channel Assignments

DMA Channel	Peripheral Assigned
0	USB Endpoint 1 Receive
1	USB Endpoint 1 Transmit
2	USB Endpoint 2 Receive
3	USB Endpoint 2 Transmit
4	USB Endpoint 3 Receive
5	USB Endpoint 3 Transmit
8	UART0 Receive
9	UART0 Transmit
10	SSI0 Receive
11	SSI0 Transmit
22	UART1 Receive
23	UART1 Transmit
24	SSI1 Receive
25	SSI1 Transmit
30	Dedicated for software use

9.2.2 Priority

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

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

9.2.3 Arbitration Size

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

If a lower priority DMA channel uses a large arbitration size, the latency for higher priority channels will be increased because the µDMA controller will complete the lower priority burst before checking for higher priority requests. Therefore, lower priority channels should not use a large arbitration size for best response on high priority channels.

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

9.2.4 Request Types

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

The µDMA controller responds differently depending on whether the peripheral is making a single request or a burst request. If both are asserted and the µDMA channel has been set up for a burst transfer, then the burst request takes precedence. See Table 9-2 on page 197, which shows how each peripheral supports the two request types.

Table 9-2. Request Type Support

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

9.2.4.1 Single Request

When a single request is detected, and not a burst request, the µDMA controller will transfer one item, and then stop and wait for another request.

9.2.4.2 Burst Request

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

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

9.2.5 Channel Configuration

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

Table 9-3 on page 198 shows the layout in memory of the channel control table. Each channel may have one or two control structures in the contol table: a primary control structure and an optional alternate control structure. The table is organized so that all of the primary entries are in the first half of the table and all the alternate structures are in the second half of the table. The primary entry is used for simple transfer modes where transfers can be reconfigured and restarted after each transfer is complete. In this case, the alternate control structures are not used and therefore only the first half of the table needs to be allocated in memory. The second half of the control table is not needed and that memory can be used for something else. If a more complex transfer mode is used such as ping-pong or scatter-gather, then the alternate control structure is also used and memory space should be allocated for the entire table.

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

Table 9-3. Control Structure Memory Map

Offset	Channel
0x0	0, Primary
0x10	1, Primary
...	...
0x1F0	31, Primary
0x200	0, Alternate
0x210	1, Alternate
...	...
0x3F0	31, Alternate

Table 9-4 on page 198 shows an individual control structure entry in the control table. Each entry has a source and destination *end* pointer. These pointers point to the ending address of the transfer and are inclusive. If the source or destination is non-incrementing (as for a peripheral register), then the pointer should point to the transfer address.

Table 9-4. Channel Control Structure

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

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

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

- Transfer mode

The control word and each field are described in detail in “µDMA Channel Control Structure” on page 215. The µDMA controller updates the transfer size and transfer mode fields as the transfer is performed. At the end of a transfer, the transfer size will indicate 0, and the transfer mode will indicate "stopped". Since the control word is modified by the µDMA controller, it must be reconfigured before each new transfer. The source and destination end pointers are not modified so they can be left unchanged if the source or destination addresses remain the same.

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

9.2.6 Transfer Modes

The µDMA controller supports several transfer modes. Two of the modes support simple one-time transfers. There are several complex modes that are meant to support a continuous flow of data.

9.2.6.1 Stop Mode

While Stop is not actually a transfer mode, it is a valid value for the mode field of the control word. When the mode field has this value, the µDMA controller will not perform a transfer and will disable the channel if it is enabled. At the end of a transfer, the µDMA controller will update the control word to set the mode to Stop.

9.2.6.2 Basic Mode

In Basic mode, the µDMA controller will perform transfers as long as there are more items to transfer and a transfer request is present. This mode is used with peripherals that assert a DMA request signal whenever the peripheral is ready for a data transfer. Basic mode should not be used in any situation where the request is momentary but the entire transfer should be completed. For example, for a software initiated transfer, the request is momentary, and if Basic mode is used then only one item will be transferred on a software request.

When all of the items have been transferred using Basic mode, the µDMA controller will set the mode for that channel to Stop.

9.2.6.3 Auto Mode

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

When all the items have been transferred using Auto mode, the µDMA controller will set the mode for that channel to Stop.

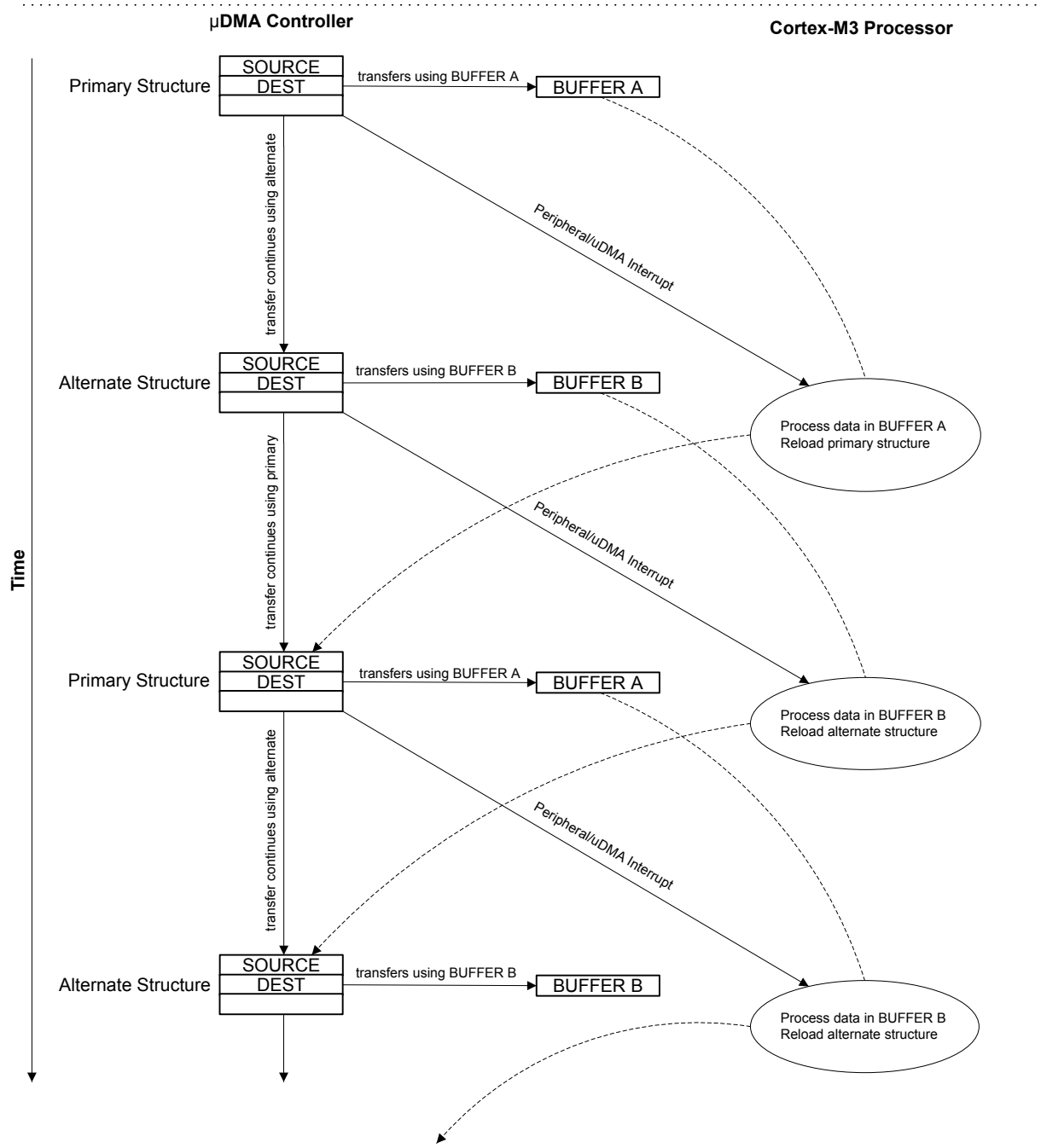
9.2.6.4 Ping-Pong

Ping-Pong mode is used to support a continuous data flow to or from a peripheral. To use Ping-Pong mode, both the primary and alternate data structures are used. Both are set up by the processor for data transfer between memory and a peripheral. Then the transfer is started using the primary control structure. When the transfer using the primary control structure is complete, the µDMA controller will then read the alternate control structure for that channel to continue the transfer. Each time this happens, an interrupt is generated and the processor can reload the control structure for the just-completed transfer. Data flow can continue indefinitely this way, using the primary and

alternate control structures to switch back and forth between buffers as the data flows to or from the peripheral.

Refer to Figure 9-2 on page 200 for an example showing operation in Ping-Pong mode.

Figure 9-2. Example of Ping-Pong DMA Transaction



9.2.6.5 Memory Scatter-Gather

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

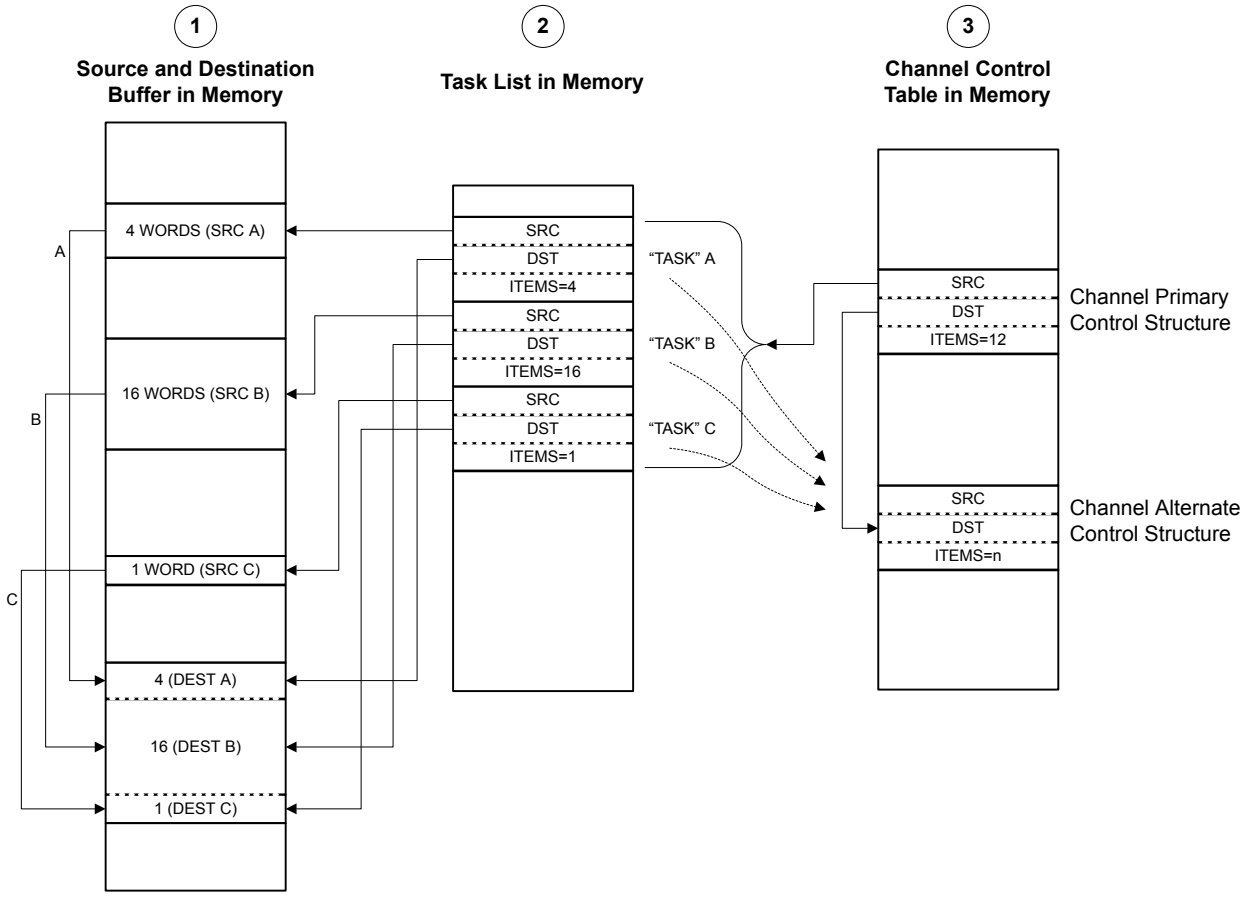
In Memory Scatter-Gather mode, the primary control structure is used to program the alternate control structure from a table in memory. The table is set up by the processor software and contains a list of control structures, each containing the source and destination end pointers, and the control word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

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

Refer to Figure 9-3 on page 202 and Figure 9-4 on page 203, which show an example of operation in Memory Scatter-Gather mode. This example shows a *gather* operation, where data in three separate buffers in memory will be copied together into one buffer. Figure 9-3 on page 202 shows how the application sets up a µDMA *task list* in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that will be used for the operation is configured to copy from the task list to the alternate control structure.

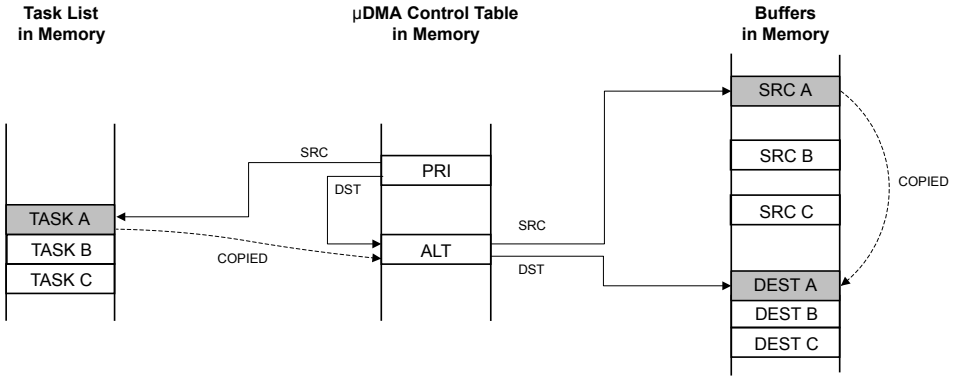
Figure 9-4 on page 203 shows the sequence as the µDMA controller peforms the three sets of copy operations. First, using the primary control structure, the µDMA controller loads the alternate control structure with task A. It then peforms the copy operation specified by task A, copying the data from the source buffer A to the destination buffer. Next, the µDMA controller again uses the primary control structure to load task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for task C.

Figure 9-3. Memory Scatter-Gather, Setup and Configuration



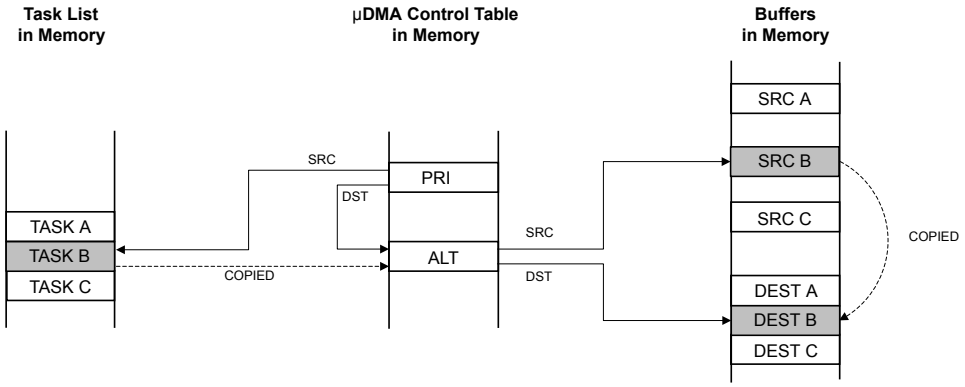
NOTES:
1. Application has a need to copy data items from three separate location in memory into one combined buffer.
2. Application sets up uDMA "task list" in memory, which contains the pointers and control configuration for three uDMA copy "tasks."
3. Application sets up the channel primary control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

Figure 9-4. Memory Scatter-Gather, µDMA Copy Sequence



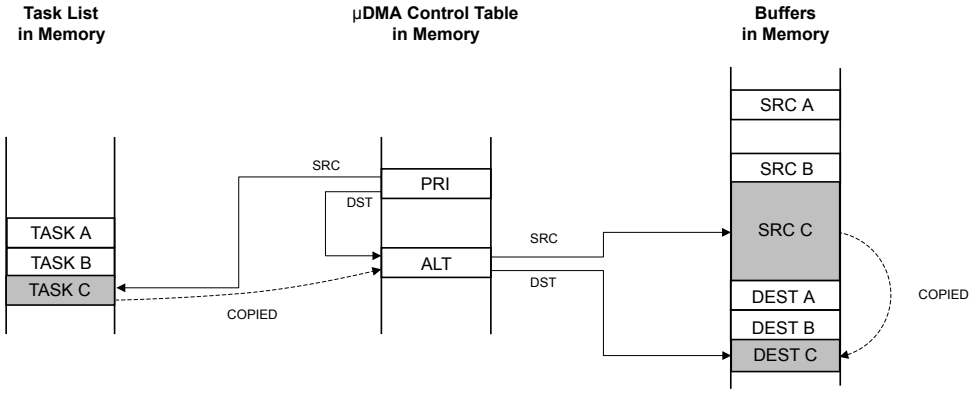
Using the channel's primary control structure, the µDMA controller copies task A configuration to the channel's alternate control structure.

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



Using the channel's primary control structure, the µDMA controller copies task B configuration to the channel's alternate control structure.

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



Using the channel's primary control structure, the µDMA controller copies task C configuration to the channel's alternate control structure.

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

9.2.6.6 Peripheral Scatter-Gather

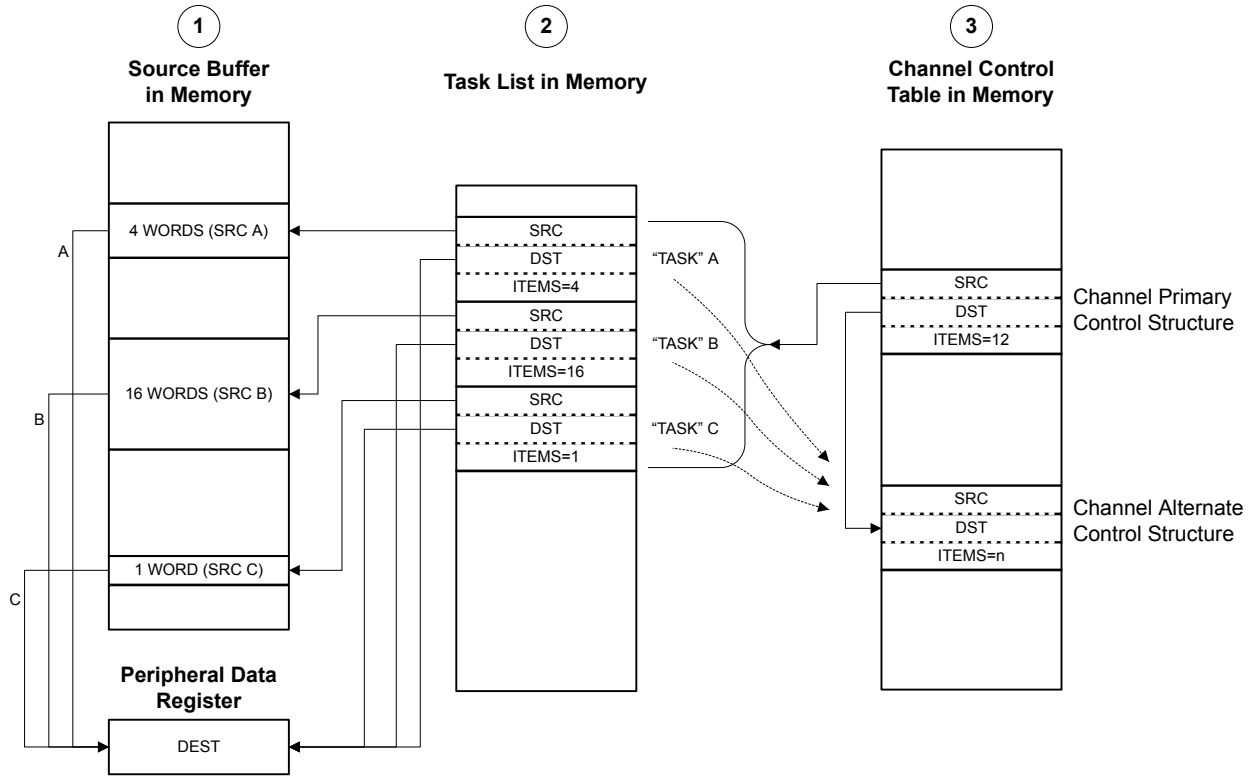
Peripheral Scatter-Gather mode is very similar to Memory Scatter-Gather, except that the transfers are controlled by a peripheral making a DMA request. Upon detecting a DMA request from the peripheral, the µDMA controller will use the primary control structure to copy one entry from the list to the alternate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

By programming the µDMA controller using this method, data can be transferred to or from a peripheral from a set of arbitrary locations whenever the peripheral is ready to transfer data.

Refer to Figure 9-5 on page 205 and Figure 9-6 on page 206, which show an example of operation in Peripheral Scatter-Gather mode. This example shows a gather operation, where data from three separate buffers in memory will be copied to a single peripheral data register. Figure 9-5 on page 205 shows how the application sets up a µDMA task list in memory that is used by the controller to perform three sets of copy operations from different locations in memory. The primary control structure for the channel that will be used for the operation is configured to copy from the task list to the alternate control structure.

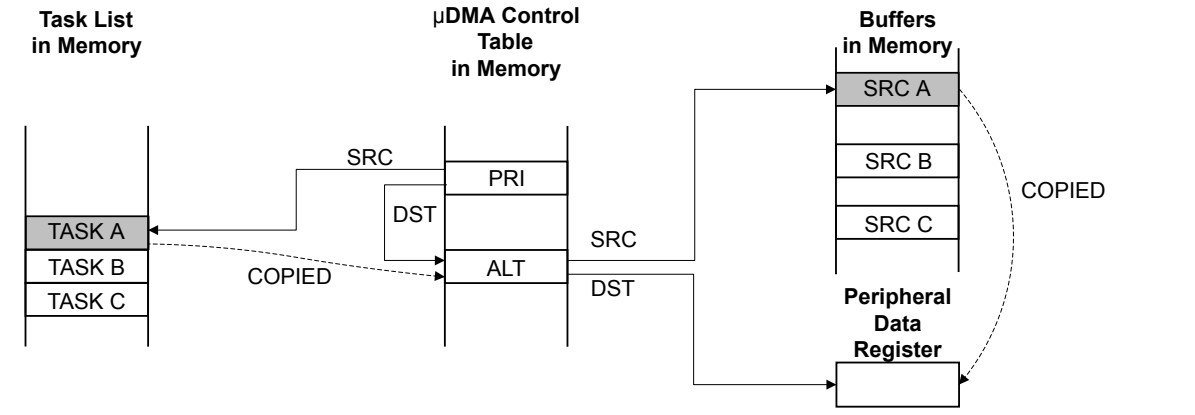
Figure 9-6 on page 206 shows the sequence as the µDMA controller peforms the three sets of copy operations. First, using the primary control structure, the µDMA controller loads the alternate control structure with task A. It then peforms the copy operation specified by task A, copying the data from the source buffer A to the peripheral data register. Next, the µDMA controller again uses the primary control structure to load task B into the alternate control structure, and then performs the B operation with the alternate control structure. The process is repeated for task C.

Figure 9-5. Peripheral Scatter-Gather, Setup and Configuration



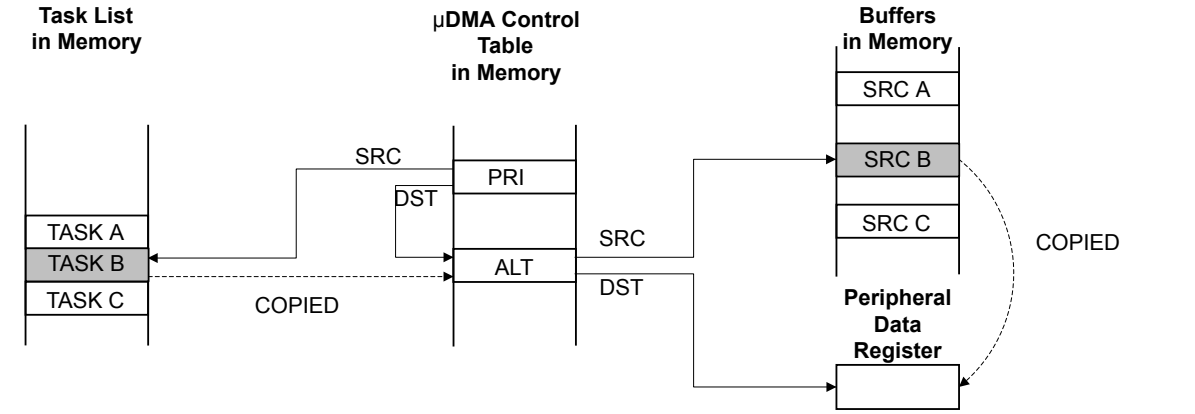
NOTES:
1. Application has a need to copy data items from three separate location in memory into a peripheral data register.
2. Application sets up µDMA "task list" in memory, which contains the pointers and control configuration for three uDMA copy "tasks."
3. Application sets up the channel primary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

Figure 9-6. Peripheral Scatter-Gather, µDMA Copy Sequence



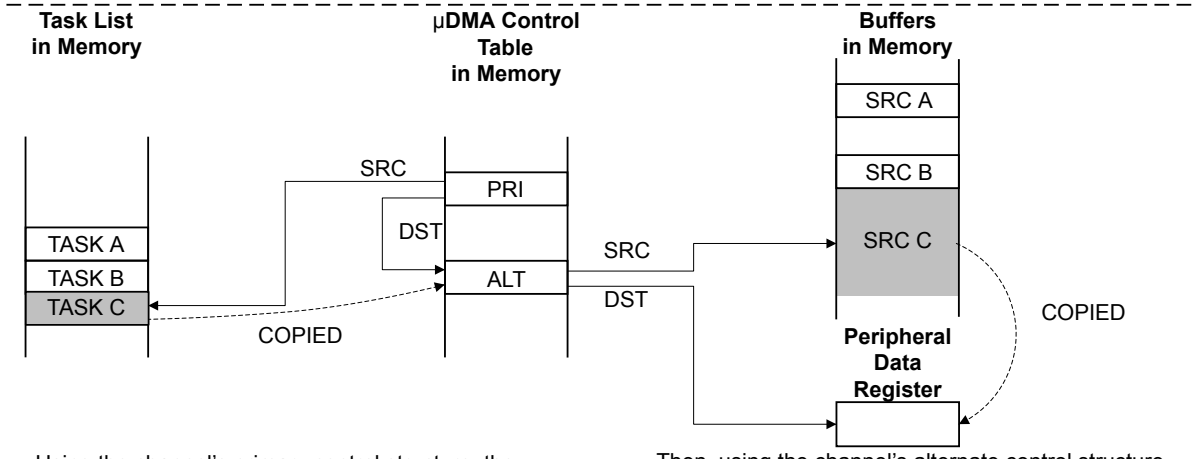
Using the channel's primary control structure, the µDMA controller copies task A configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the µDMA controller copies data from the source buffer A to the peripheral data register.



Using the channel's primary control structure, the µDMA controller copies task B configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the µDMA controller copies data from the source buffer B to the peripheral data register.



Using the channel's primary control structure, the µDMA controller copies task C configuration to the channel's alternate control structure.

Then, using the channel's alternate control structure, the µDMA controller copies data from the source buffer C to the peripheral data register.

9.2.7 Transfer Size and Increment

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

Table 9-5 on page 207 shows the configuration to read from a peripheral that supplies 8-bit data.

Table 9-5. µDMA Read Example: 8-Bit Peripheral

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

9.2.8 Peripheral Interface

Each peripheral that supports µDMA has a DMA single request and/or burst request signal that is asserted when the device is ready to transfer data. The request signal can be disabled or enabled by using the **DMA Channel Request Mask Set (DMAREQMASKSET)** and **DMA Channel Request Mask Clear (DMAREQMASKCLR)** registers. The DMA request signal is disabled, or masked, when the channel request mask bit is set. When the request is not masked, the DMA channel is configured correctly and enabled, and the peripheral asserts the DMA request signal, the µDMA controller will begin the transfer.

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

The interrupt request from the µDMA controller is automatically cleared when the interrupt handler is activated.

9.2.9 Software Request

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

It is possible to initiate a transfer on any channel using the **DMASWREQ** register. If a request is initiated by software using a peripheral DMA channel, then the completion interrupt will occur on the interrupt vector for the peripheral instead of the software interrupt vector. This means that any

channel may be used for software requests as long as the corresponding peripheral is not using µDMA.

9.2.10 Interrupts and Errors

When a DMA transfer is complete, the µDMA controller will generate a completion interrupt on the interrupt vector of the peripheral. If the transfer uses the software DMA channel, then the completion interrupt will occur on the dedicated software DMA interrupt vector.

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

If the peripheral generates an error that causes an interrupt, the interrupt will be generated on the interrupt vector for that peripheral. This is the same whether or not µDMA is being used with the peripheral.

Table 9-6 on page 208 shows the dedicated interrupt assignments for the µDMA controller.

Table 9-6. µDMA Interrupt Assignments

Interrupt	Assignment
46	µDMA Software Channel Transfer
47	µDMA Error

9.3 Initialization and Configuration

9.3.1 Module Initialization

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

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

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

9.3.2 Configuring a Memory-to-Memory Transfer

µDMA channel 30 is dedicated for software-initiated transfers. However, any channel can be used for software-initiated, memory-to-memory transfer if the associated peripheral is not being used.

9.3.2.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Set bit 30 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.

2. Set bit 30 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.

3. Set bit 30 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the µDMA controller to respond to single and burst requests.

4. Set bit 30 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the µDMA controller to recognize requests for this channel.

9.3.2.2 Configure the Channel Control Structure

Now the channel control structure must be configured.

This example will transfer 256 32-bit words from one memory buffer to another. Channel 30 is used for a software transfer, and the control structure for channel 30 is at offset 0x1E0 of the channel control table. The channel control structure for channel 30 is located at the offsets shown in Table 9-7 on page 209.

Table 9-7. Channel Control Structure Offsets for Channel 30

Offset	Description
Control Table Base + 0x1E0	Channel 30 Source End Pointer
Control Table Base + 0x1E4	Channel 30 Destination End Pointer
Control Table Base + 0x1E8	Channel 30 Control Word

Configure the Source and Destination

The source and destination end pointers must be set to the last address for the transfer (inclusive).

1. Set the source end pointer at offset 0x1E0 to the address of the source buffer + 0x3FC.

2. Set the destination end pointer at offset 0x1E4 to the address of the destination buffer + 0x3FC.

The control word at offset 0x1E8 must be programmed according to Table 9-8 on page 209.

Table 9-8. Channel Control Word Configuration for Memory Transfer Example

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	2	32-bit destination address increment
DSTSIZE	29:28	2	32-bit destination data size
SRCINC	27:26	2	32-bit source address increment
SRCSIZE	25:24	2	32-bit source data size
reserved	23:18	0	Reserved
ARBSIZE	17:14	3	Arbitrates after 8 transfers
XFERSIZE	13:4	255	Transfer 256 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	2	Use Auto-request transfer mode

9.3.2.3 Start the Transfer

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 30 of the **DMA Channel Enable Set (DMAENASET)** register.

2. Issue a transfer request by setting bit 30 of the **DMA Channel Software Request (DMASWREQ)** register.

The DMA transfer will now take place. If the interrupt is enabled, then the processor will be notified by interrupt when the transfer is complete. If needed, the status can be checked by reading bit 30 of the **DMAENASET** register. This bit will be automatically cleared when the transfer is complete. The status can also be checked by reading the XFERMODE field of the channel control word at offset 0x1E8. This field will automatically be set to 0 at the end of the transfer.

9.3.3 Configuring a Peripheral for Simple Transmit

This example will set up the µDMA controller to transmit a buffer of data to a peripheral. The peripheral has a transmit FIFO with a trigger level of 4. The example peripheral will use µDMA channel 7.

9.3.3.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Set bit 7 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.
2. Set bit 7 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.
3. Set bit 7 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the µDMA controller to respond to single and burst requests.
4. Set bit 7 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the µDMA controller to recognize requests for this channel.

9.3.3.2 Configure the Channel Control Structure

Now the channel control structure must be configured. This example will transfer 64 8-bit bytes from a memory buffer to the peripheral's transmit FIFO register. This example uses µDMA channel 7, and the control structure for channel 7 is at offset 0x070 of the channel control table. The channel control structure for channel 7 is located at the offsets shown in Table 9-9 on page 210.

Table 9-9. Channel Control Structure Offsets for Channel 7

Offset	Description
Control Table Base + 0x070	Channel 7 Source End Pointer
Control Table Base + 0x074	Channel 7 Destination End Pointer
Control Table Base + 0x078	Channel 7 Control Word

Configure the Source and Destination

The source and destination end pointers must be set to the last address for the transfer (inclusive). Since the peripheral pointer does not change, it simply points to the peripheral's data register.

1. Set the source end pointer at offset 0x070 to the address of the source buffer + 0x3F.
2. Set the destination end pointer at offset 0x074 to the address of the peripheral's transmit FIFO register.

The control word at offset 0x078 must be programmed according to Table 9-10 on page 211.

Table 9-10. Channel Control Word Configuration for Peripheral Transmit Example

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	3	Destination address does not increment
DSTSIZE	29:28	0	8-bit destination data size
SRCINC	27:26	0	8-bit source address increment
SRCSIZE	25:24	0	8-bit source data size
reserved	23:18	0	Reserved
ARBSIZE	17:14	2	Arbitrates after 4 transfers
XFERSIZE	13:4	63	Transfer 64 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	1	Use Basic transfer mode

Note: In this example, it is not important if the peripheral makes a single request or a burst request. Since the peripheral has a FIFO that will trigger at a level of 4, the arbitration size is set to 4. If the peripheral does make a burst request, then 4 bytes will be transferred, which is what the FIFO can accomodate. If the peripheral makes a single request (if there is any space in the FIFO), then one byte will be transferred at a time. If it is important to the application that transfers only be made in bursts, then the channel useburst SET[n] bit should be set by writing a 1 to bit 7 of the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register.

9.3.3.3 Start the Transfer

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 7 of the **DMA Channel Enable Set (DMAENASET)** register.

The µDMA controller is now configured for transfer on channel 7. The controller will make transfers to the peripheral whenever the peripheral asserts a DMA request. The transfers will continue until the entire buffer of 64 bytes has been transferred. When that happens, the µDMA controller will disable the channel and set the XFERMODE field of the channel control word to 0 (Stopped). The status of the transfer can be checked by reading bit 7 of the **DMA Channel Enable Set (DMAENASET)** register. This bit will be automatically cleared when the transfer is complete. The status can also be checked by reading the XFERMODE field of the channel control word at offset 0x078. This field will automatically be set to 0 at the end of the transfer.

If peripheral interrupts were enabled, then the peripheral interrupt handler would receive an interrupt when the entire transfer was complete.

9.3.4 Configuring a Peripheral for Ping-Pong Receive

This example will set up the µDMA controller to continuously receive 8-bit data from a peripheral into a pair of 64 byte buffers. The peripheral has a receive FIFO with a trigger level of 8. The example peripheral will use µDMA channel 8.

9.3.4.1 Configure the Channel Attributes

First, configure the channel attributes:

1. Set bit 8 of the **DMA Channel Priority Set (DMAPRIOSET)** or **DMA Channel Priority Clear (DMAPRIOCLR)** registers to set the channel to High priority or Default priority.

2. Set bit 8 of the **DMA Channel Primary Alternate Clear (DMAALTCLR)** register to select the primary channel control structure for this transfer.

3. Set bit 8 of the **DMA Channel Useburst Clear (DMAUSEBURSTCLR)** register to allow the µDMA controller to respond to single and burst requests.

4. Set bit 8 of the **DMA Channel Request Mask Clear (DMAREQMASKCLR)** register to allow the µDMA controller to recognize requests for this channel.

9.3.4.2 Configure the Channel Control Structure

Now the channel control structure must be configured. This example will transfer 8-bit bytes from the peripheral's receive FIFO register into two memory buffers of 64 bytes each. As data is received, when one buffer is full, the µDMA controller switches to use the other.

To use Ping-Pong buffering, both primary and alternate channel control structures must be used. The primary control structure for channel 8 is at offset 0x080 of the channel control table, and the alternate channel control structure is at offset 0x280. The channel control structures for channel 8 are located at the offsets shown in Table 9-11 on page 212.

Table 9-11. Primary and Alternate Channel Control Structure Offsets for Channel 8

Offset	Description
Control Table Base + 0x080	Channel 8 Primary Source End Pointer
Control Table Base + 0x084	Channel 8 Primary Destination End Pointer
Control Table Base + 0x088	Channel 8 Primary Control Word
Control Table Base + 0x280	Channel 8 Alternate Source End Pointer
Control Table Base + 0x284	Channel 8 Alternate Destination End Pointer
Control Table Base + 0x288	Channel 8 Alternate Control Word

Configure the Source and Destination

The source and destination end pointers must be set to the last address for the transfer (inclusive). Since the peripheral pointer does not change, it simply points to the peripheral's data register. Both the primary and alternate sets of pointers must be configured.

1. Set the primary source end pointer at offset 0x080 to the address of the peripheral's receive buffer.

2. Set the primary destination end pointer at offset 0x084 to the address of ping-pong buffer A + 0x3F.

3. Set the alternate source end pointer at offset 0x280 to the address of the peripheral's receive buffer.

4. Set the alternate destination end pointer at offset 0x284 to the address of ping-pong buffer B + 0x3F.

The primary control word at offset 0x088, and the alternate control word at offset 0x288 must be programmed according to Table 9-10 on page 211. Both control words are initially programmed the same way.

1. Program the primary channel control word at offset 0x088 according to Table 9-12 on page 213.

2. Program the alternate channel control word at offset 0x288 according to Table 9-12 on page 213.

Table 9-12. Channel Control Word Configuration for Peripheral Ping-Pong Receive Example

Field in DMACHCTL	Bits	Value	Description
DSTINC	31:30	0	8-bit destination address increment
DSTSIZE	29:28	0	8-bit destination data size
SRCINC	27:26	3	Source address does not increment
SRCSIZE	25:24	0	8-bit source data size
reserved	23:18	0	Reserved
ARBSIZE	17:14	3	Arbitrates after 8 transfers
XFERSIZE	13:4	63	Transfer 64 items
NXTUSEBURST	3	0	N/A for this transfer type
XFERMODE	2:0	3	Use Ping-Pong transfer mode

Note: In this example, it is not important if the peripheral makes a single request or a burst request. Since the peripheral has a FIFO that will trigger at a level of 8, the arbitration size is set to 8. If the peripheral does make a burst request, then 8 bytes will be transferred, which is what the FIFO can accomodate. If the peripheral makes a single request (if there is any data in the FIFO), then one byte will be transferred at a time. If it is important to the application that transfers only be made in bursts, then the channel useburst SET[n] bit should be set by writing a 1 to bit 8 of the **DMA Channel Useburst Set (DMAUSEBURSTSET)** register.

9.3.4.3 Configure the Peripheral Interrupt

In order to use µDMA Ping-Pong mode, it is best to use an interrupt handler. (It is also possible to use ping-pong mode without interrupts by polling). The interrupt handler will be triggered after each buffer is complete.

1. Configure and enable an interrupt handler for the peripheral.

9.3.4.4 Enable the µDMA Channel

Now the channel is configured and is ready to start.

1. Enable the channel by setting bit 8 of the **DMA Channel Enable Set (DMAENASET)** register.

9.3.4.5 Process Interrupts

The µDMA controller is now configured and enabled for transfer on channel 8. When the peripheral asserts the DMA request signal, the µDMA controller will make transfers into buffer A using the primary channel control structure. When the primary transfer to buffer A is complete, it will switch to the alternate channel control structure and make transfers into buffer B. At the same time, the primary channel control word mode field will be set to indicate Stopped, and an interrupt will be triggered.

When an interrupt is triggered, the interrupt handler must determine which buffer is complete and process the data, or set a flag that the data needs to be processed by non-interrupt buffer processing code. Then the next buffer transfer must be set up.

In the interrupt handler:

1. Read the primary channel control word at offset 0x088 and check the XFERMODE field. If the field is 0, this means buffer A is complete. If buffer A is complete, then:

a. Process the newly received data in buffer A, or signal the buffer processing code that buffer A has data available.

b. Reprogram the primary channel control word at offset 0x88 according to Table 9-12 on page 213.

2. Read the alternate channel control word at offset 0x288 and check the XFERMODE field. If the field is 0, this means buffer B is complete. If buffer B is complete, then:

a. Process the newly received data in buffer B, or signal the buffer processing code that buffer B has data available.

b. Reprogram the alternate channel control word at offset 0x288 according to Table 9-12 on page 213.

9.4 Register Map

Table 9-13 on page 214 lists the µDMA channel control structures and registers. The channel control structure shows the layout of one entry in the channel control table. The channel control table is located in system memory, and the location is determined by the application, that is, the base address is n/a (not applicable). In the table below, the offset for the channel control structures is the offset from the entry in the channel control table. See “Channel Configuration” on page 197 and Table 9-3 on page 198 for a description of how the entries in the channel control table are located in memory. The µDMA register addresses are given as a hexadecimal increment, relative to the µDMA base address of 0x400F.F000.

Table 9-13. µDMA Register Map

Offset	Name	Type	Reset	Description	See page
µDMA Channel Control Structure					
0x000	DMASRCENDP	R/W	-	DMA Channel Source Address End Pointer	216
0x004	DMADSTENDP	R/W	-	DMA Channel Destination Address End Pointer	217
0x008	DMACHCTL	R/W	-	DMA Channel Control Word	218
µDMA Registers					
0x000	DMASTAT	RO	0x001F.0000	DMA Status	222
0x004	DMACFG	WO	-	DMA Configuration	224
0x008	DMACTLBASE	R/W	0x0000.0000	DMA Channel Control Base Pointer	225
0x00C	DMAALTBASE	RO	0x0000.0200	DMA Alternate Channel Control Base Pointer	226
0x010	DMAWAITSTAT	RO	0x0000.0000	DMA Channel Wait on Request Status	227
0x014	DMASWREQ	WO	-	DMA Channel Software Request	228
0x018	DMAUSEBURSTSET	R/W	0x0000.0000	DMA Channel Useburst Set	229
0x01C	DMAUSEBURSTCLR	WO	-	DMA Channel Useburst Clear	231
0x020	DMAREQMASKSET	R/W	0x0000.0000	DMA Channel Request Mask Set	232
0x024	DMAREQMASKCLR	WO	-	DMA Channel Request Mask Clear	234

Offset	Name	Type	Reset	Description	See page
0x028	DMAENASET	R/W	0x0000.0000	DMA Channel Enable Set	235
0x02C	DMAENACLR	WO	-	DMA Channel Enable Clear	237
0x030	DMAALTSET	R/W	0x0000.0000	DMA Channel Primary Alternate Set	238
0x034	DMAALTCLR	WO	-	DMA Channel Primary Alternate Clear	240
0x038	DMAPRIOSET	R/W	0x0000.0000	DMA Channel Priority Set	241
0x03C	DMAPRIOCLR	WO	-	DMA Channel Priority Clear	243
0x04C	DMAERRCLR	R/W	0x0000.0000	DMA Bus Error Clear	244
0xFD0	DMAPeriphID4	RO	0x0000.0004	DMA Peripheral Identification 4	250
0xFE0	DMAPeriphID0	RO	0x0000.0030	DMA Peripheral Identification 0	246
0xFE4	DMAPeriphID1	RO	0x0000.00B2	DMA Peripheral Identification 1	247
0xFE8	DMAPeriphID2	RO	0x0000.000B	DMA Peripheral Identification 2	248
0xFEC	DMAPeriphID3	RO	0x0000.0000	DMA Peripheral Identification 3	249
0xFF0	DMAPCellID0	RO	0x0000.000D	DMA PrimeCell Identification 0	251
0xFF4	DMAPCellID1	RO	0x0000.00F0	DMA PrimeCell Identification 1	252
0xFF8	DMAPCellID2	RO	0x0000.0005	DMA PrimeCell Identification 2	253
0xFFC	DMAPCellID3	RO	0x0000.00B1	DMA PrimeCell Identification 3	254

9.5 µDMA Channel Control Structure

The µDMA Channel Control Structure holds the DMA transfer settings for a DMA channel. Each channel has two control structures, which are located in a table in system memory. Refer to “Channel Configuration” on page 197 for an explanation of the Channel Control Table and the Channel Control Structure.

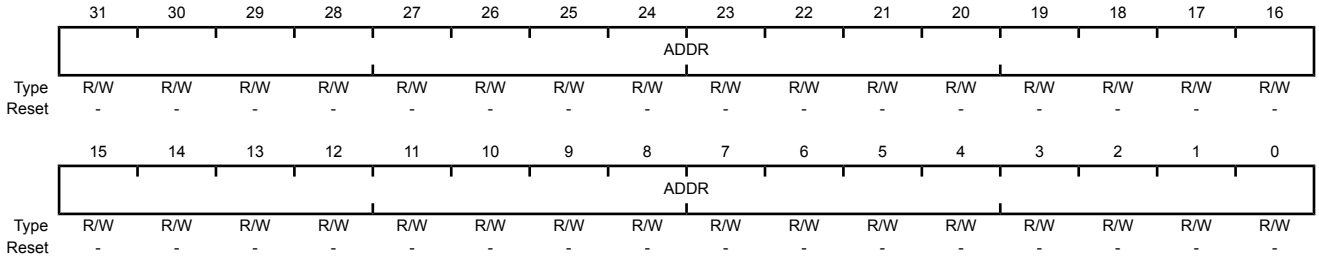
The channel control structure is one entry in the channel control table. There is a primary and alternate structure for each channel. The primary control structures are located at offsets 0x0, 0x10, 0x20 and so on. The alternate control structures are located at offsets 0x200, 0x210, 0x220, and so on.

Register 1: DMA Channel Source Address End Pointer (DMASRCENDP), offset 0x000

DMA Channel Source Address End Pointer (DMASRCENDP) is part of the Channel Control Structure, and is used to specify the source address for a DMA transfer.

DMA Channel Source Address End Pointer (DMASRCENDP)

Base n/a
Offset 0x000
Type R/W, reset -



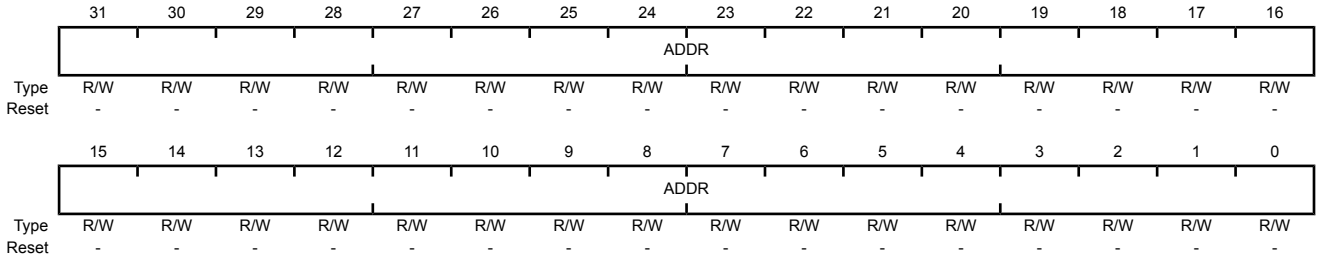
Bit/Field	Name	Type	Reset	Description
31:0	ADDR	R/W	-	Source Address End Pointer Points to the last address of the DMA transfer source (inclusive). If the source address is not incrementing, then this points at the source location itself (such as a peripheral data register).

Register 2: DMA Channel Destination Address End Pointer (DMADSTENDP), offset 0x004

DMA Channel Destination Address End Pointer (DMADSTENDP) is part of the Channel Control Structure, and is used to specify the destination address for a DMA transfer.

DMA Channel Destination Address End Pointer (DMADSTENDP)

Base n/a
Offset 0x004
Type R/W, reset -



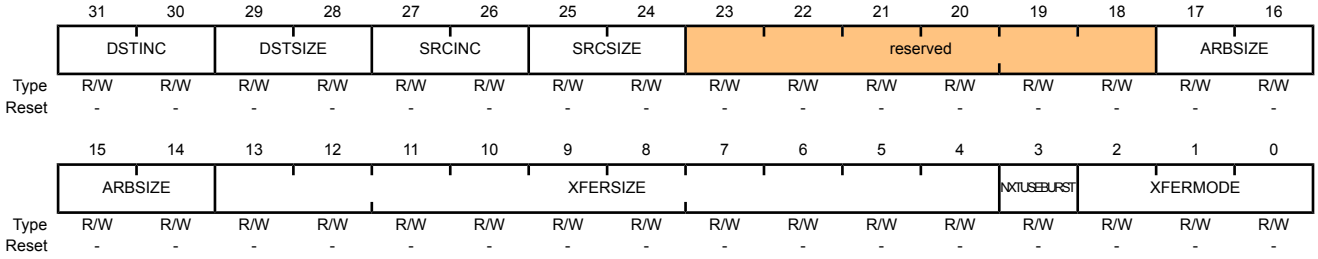
Bit/Field	Name	Type	Reset	Description
31:0	ADDR	R/W	-	Destination Address End Pointer Points to the last address of the DMA transfer destination (inclusive). If the destination address is not incrementing, then this points at the destination location itself (such as a peripheral data register).

Register 3: DMA Channel Control Word (DMACHCTL), offset 0x008

DMA Channel Control Word (DMACHCTL) is part of the Channel Control Structure, and is used to specify parameters of a DMA transfer.

DMA Channel Control Word (DMACHCTL)

Base n/a
Offset 0x008
Type R/W, reset -



Bit/Field	Name	Type	Reset	Description
31:30	DSTINC	R/W	-	Destination Address Increment Sets the bits to control the destination address increment. The address increment value must be equal or greater than the value of the destination size (DSTSIZE). Value Description 0x0 Byte Increment by 8-bit locations. 0x1 Half-word Increment by 16-bit locations. 0x2 Word Increment by 32-bit locations. 0x3 No increment Address remains set to the value of the Destination Address End Pointer (DMADSTENDP) for the channel.
29:28	DSTSIZE	R/W	-	Destination Data Size Sets the destination item data size. **Note:** You must set DSTSIZE to be the same as SRCSIZE. Value Description 0x0 Byte 8-bit data size. 0x1 Half-word 16-bit data size. 0x2 Word 32-bit data size. 0x3 Reserved

Bit/Field	Name	Type	Reset	Description
27:26	SRCINC	R/W	-	Source Address Increment Sets the bits to control the source address increment. The address increment value must be equal or greater than the value of the source size (SRCSIZE). Value Description 0x0 Byte Increment by 8-bit locations. 0x1 Half-word Increment by 16-bit locations. 0x2 Word Increment by 32-bit locations. 0x3 No increment Address remains set to the value of the Source Address End Pointer (DMASRCENDP) for the channel.
25:24	SRCSIZE	R/W	-	Source Data Size Sets the source item data size. **Note:** You must set DSTSIZE to be the same as SRCSIZE. Value Description 0x0 Byte 8-bit data size. 0x1 Half-word 16-bit data size. 0x2 Word 32-bit data size. 0x3 Reserved
23:18	reserved	R/W	-	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
17:14	ARBSIZE	R/W	-	Arbitration Size Sets the number of DMA transfers that can occur before the controller re-arbitrates. The possible arbitration rate settings represent powers of 2 and are shown below. Value Description 0x0 1 Transfer Arbitrates after each DMA transfer. 0x1 2 Transfers 0x2 4 Transfers 0x3 8 Transfers 0x4 16 Transfers 0x5 32 Transfers 0x6 64 Transfers 0x7 128 Transfers 0x8 256 Transfers 0x9 512 Transfers 0xA-0xF 1024 Transfers This means that no arbitration occurs during the DMA transfer because the maximum transfer size is 1024.
13:4	XFERSIZE	R/W	-	Transfer Size (minus 1) Sets the total number of items to transfer. The value of this field is 1 less than the number to transfer (value 0 means transfer 1 item). The maximum value for this 10-bit field is 1023 which represents a transfer size of 1024 items. The transfer size is the number of items, not the number of bytes. If the data size is 32 bits, then this value is the number of 32-bit words to transfer. The controller updates this field immediately prior to it entering the arbitration process, so it contains the number of outstanding DMA items that are necessary to complete the DMA cycle.
3	NXTUSEBURST	R/W	-	Next Useburst Controls whether the useburst SET[n] bit is automatically set for the last transfer of a peripheral scatter-gather operation. Normally, for the last transfer, if the number of remaining items to transfer is less than the arbitration size, the controller will use single transfers to complete the transaction. If this bit is set, then the controller will only use a burst transfer to complete the last transfer.

Bit/Field	Name	Type	Reset	Description
2:0	XFERMODE	R/W	-	DMA Transfer Mode Since this register is in system RAM, it has no reset value. Therefore, this field should be initialized to 0 before the channel is enabled. The operating mode of the DMA cycle. Refer to "Transfer Modes" on page 199 for a detailed explanation of transfer modes. Value Description 0x0 Stop Channel is stopped, or configuration data is invalid. 0x1 Basic The controller must receive a new request, prior to it entering the arbitration process, to enable the DMA cycle to complete. 0x2 Auto-Request The initial request (software- or peripheral-initiated) is sufficient to complete the entire transfer of XFERSIZE items without any further requests. 0x3 Ping-Pong The controller performs a DMA cycle using one of the channel control structures. After the DMA cycle completes, it performs a DMA cycle using the other channel control structure. After the next DMA cycle completes (and provided that the host processor has updated the original channel control data structure), it performs a DMA cycle using the original channel control data structure. The controller continues to perform DMA cycles until it either reads an invalid data structure or the host processor changes this field to 0x1 or 0x2. See "Ping-Pong" on page 199. 0x4 Memory Scatter-Gather When the controller operates in Memory Scatter-Gather mode, you must only use this value in the primary channel control data structure. See "Memory Scatter-Gather" on page 201. 0x5 Alternate Memory Scatter-Gather When the controller operates in Memory Scatter-Gather mode, you must only use this value in the alternate channel control data structure. 0x6 Peripheral Scatter-Gather When the controller operates in Peripheral Scatter-Gather mode, you must only use this value in the primary channel control data structure. See "Peripheral Scatter-Gather" on page 204. 0x7 Alternate Peripheral Scatter-Gather When the controller operates in Peripheral Scatter-Gather mode, you must only use this value in the alternate channel control data structure.

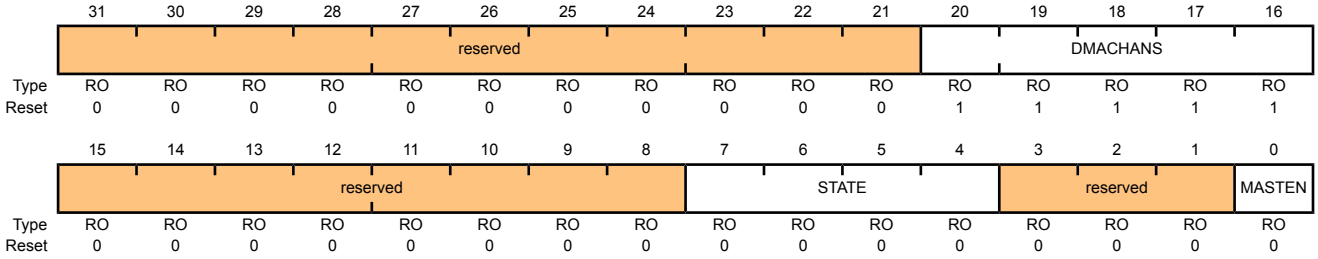
9.6 µDMA Register Descriptions

The register addresses given are relative to the µDMA base address of 0x400F.F000.

Register 4: DMA Status (DMASTAT), offset 0x000

The **DMA Status (DMASTAT)** register returns the status of the controller. You cannot read this register when the controller is in the reset state.

DMA Status (DMASTAT)
Base 0x400F.F000
Offset 0x000
Type RO, reset 0x001F.0000



Bit/Field	Name	Type	Reset	Description
31:21	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
20:16	DMACHANS	RO	0x1F	Available DMA Channels Minus 1 This bit contains a value equal to the number of DMA channels the controller is configured to use, minus one. That is, 32 DMA channels.
15:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

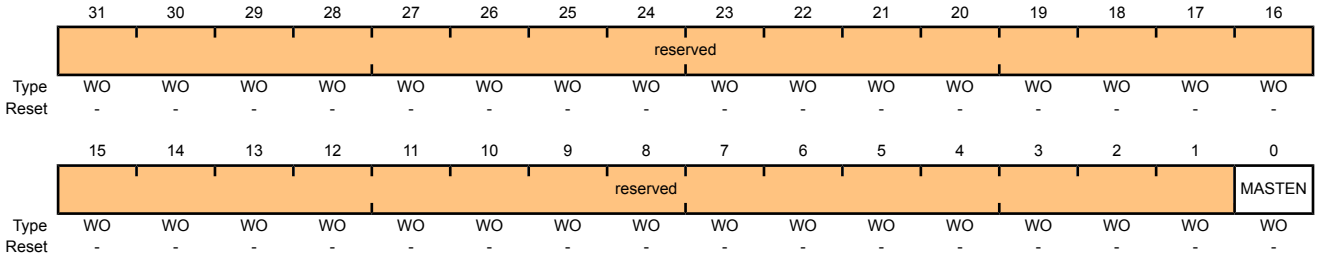
Bit/Field	Name	Type	Reset	Description
7:4	STATE	RO	0x00	Control State Machine State Current state of the control state machine. State can be one of the following. Value Description 0x0 Idle 0x1 Read Chan Control Data Reading channel controller data. 0x2 Read Source End Ptr Reading source end pointer. 0x3 Read Dest End Ptr Reading destination end pointer. 0x4 Read Source Data Reading source data. 0x5 Write Dest Data Writing destination data. 0x6 Wait for Req Clear Waiting for DMA request to clear. 0x7 Write Chan Control Data Writing channel controller data. 0x8 Stalled 0x9 Done 0xA-0xF Undefined
3:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	MASTEN	RO	0x00	Master Enable Returns status of the controller. Value Description 0 Disabled 1 Enabled

Register 5: DMA Configuration (DMACFG), offset 0x004

The **DMACFG** register controls the configuration of the controller.

DMA Configuration (DMACFG)

Base 0x400F.F000
Offset 0x004
Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:1	reserved	WO	-	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	MASTEN	WO	-	Controller Master Enable Enables the controller. Value Description 0 Disables 1 Enables

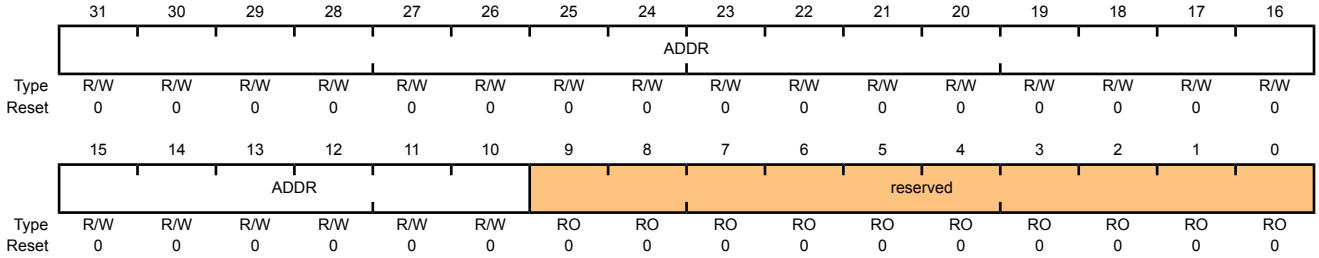
Register 6: DMA Channel Control Base Pointer (DMACTLBASE), offset 0x008

The **DMACTLBASE** register must be configured so that the base pointer points to a location in system memory.

The amount of system memory that you must assign to the controller depends on the number of DMA channels used and whether you configure it to use the alternate channel control data structure. See “Channel Configuration” on page 197 for details about the Channel Control Table. The base address must be aligned on a 1024-byte boundary. You cannot read this register when the controller is in the reset state.

DMA Channel Control Base Pointer (DMACTLBASE)

Base 0x400F.F000
Offset 0x008
Type R/W, reset 0x0000.0000



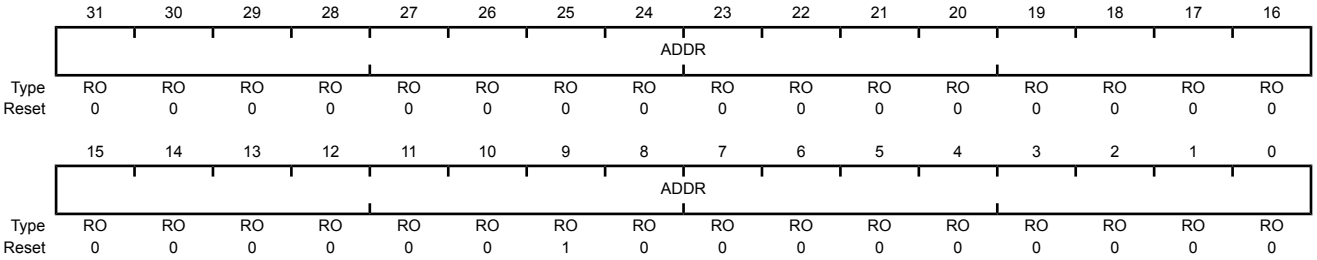
Bit/Field	Name	Type	Reset	Description
31:10	ADDR	R/W	0x00	Channel Control Base Address Pointer to the base address of the channel control table. The base address must be 1024-byte aligned.
9:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 7: DMA Alternate Channel Control Base Pointer (DMAALTBASE), offset 0x00C

The **DMAALTBASE** register returns the base address of the alternate channel control data. This register removes the necessity for application software to calculate the base address of the alternate channel control structures. You cannot read this register when the controller is in the reset state.

DMA Alternate Channel Control Base Pointer (DMAALTBASE)

Base 0x400F.F000
Offset 0x00C
Type RO, reset 0x0000.0200



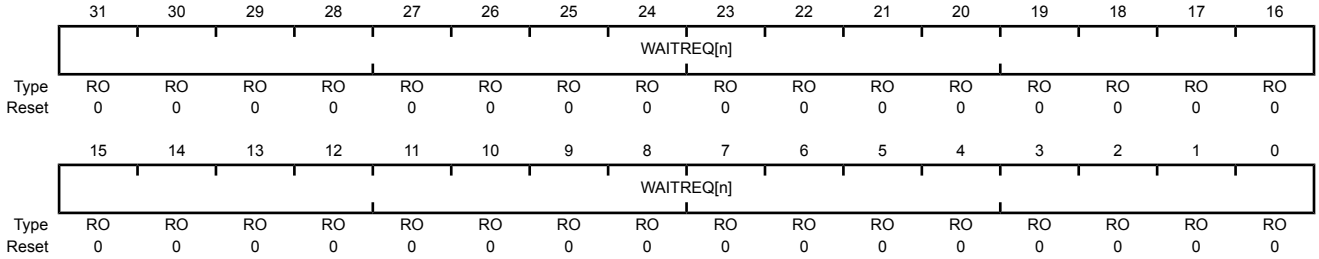
Bit/Field	Name	Type	Reset	Description
31:0	ADDR	RO	0x200	Alternate Channel Address Pointer Provides the base address of the alternate channel control structures.

Register 8: DMA Channel Wait on Request Status (DMAWAITSTAT), offset 0x010

This read-only register indicates that the µDMA channel is waiting on a request. A peripheral can pull this Low to hold off the µDMA from performing a single request until the peripheral is ready for a burst request. The use of this feature is dependent on the design of the peripheral and is used to enhance performance of the µDMA with that peripheral. You cannot read this register when the controller is in the reset state.

DMA Channel Wait on Request Status (DMAWAITSTAT)

Base 0x400F.F000
Offset 0x010
Type RO, reset 0x0000.0000



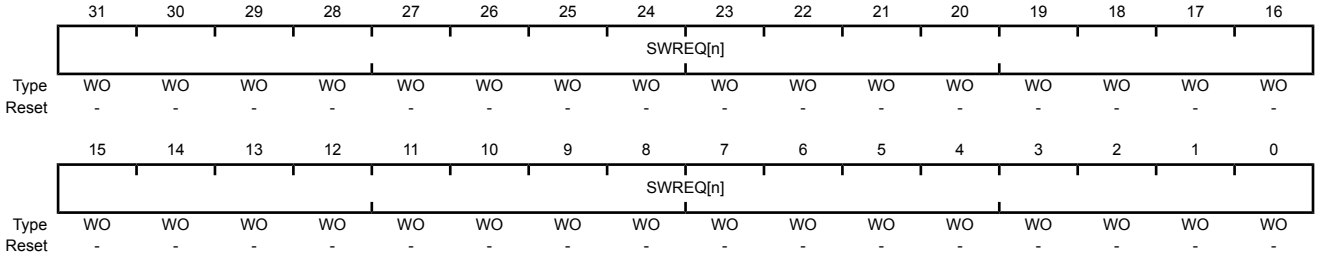
Bit/Field	Name	Type	Reset	Description
31:0	WAITREQ[n]	RO	0x00	Channel [n] Wait Status Channel wait on request status. For each channel 0 through 31, a 1 in the corresponding bit field indicates that the channel is waiting on a request.

Register 9: DMA Channel Software Request (DMASWREQ), offset 0x014

Each bit of the **DMASWREQ** register represents the corresponding DMA channel. When you set a bit, it generates a request for the specified DMA channel.

DMA Channel Software Request (DMASWREQ)

Base 0x400F.F000
Offset 0x014
Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	SWREQ[n]	WO	-	Channel [n] Software Request For each channel 0 through 31, write a 1 to the corresponding bit field to generate a software DMA request for that DMA channel. Writing a 0 does not create a DMA request for the corresponding channel.

Register 10: DMA Channel Useburst Set (DMAUSEBURSTSET), offset 0x018

Each bit of the **DMAUSEBURSTSET** register represents the corresponding DMA channel. Writing a 1 disables the peripheral's single request input from generating requests, and therefore only the peripheral's burst request generates requests. Reading the register returns the status of useburst.

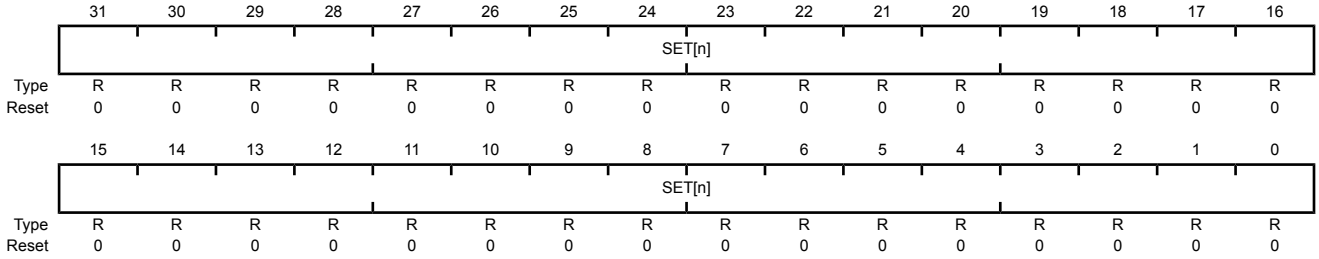
When there are fewer items remaining to transfer than the arbitration (burst) size, the controller automatically clears the useburst bit to 0. This enables the remaining items to transfer using single requests. This bit should not be set for a peripheral's channel that does not support the burst request model.

Refer to “Request Types” on page 197 for more details about request types.

Reads

DMA Channel Useburst Set (DMAUSEBURSTSET)

Base 0x400F.F000
Offset 0x018
Type RO, reset 0x0000.0000

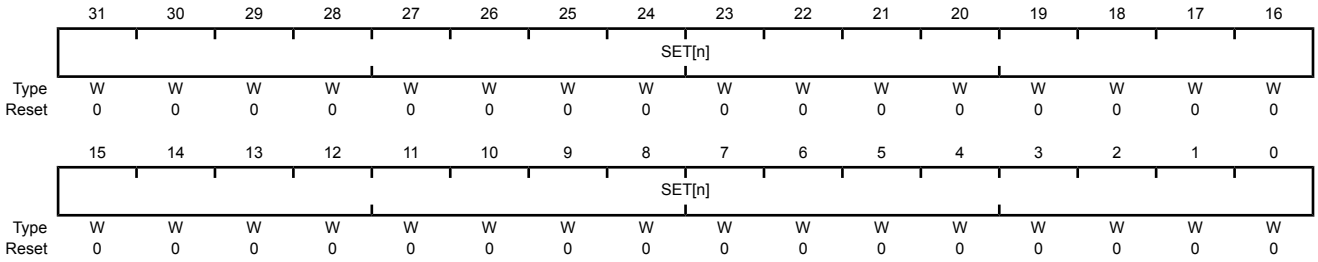


Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R	0x00	Channel [n] Useburst Status Returns the useburst status of channel [n]. Value Description 0 Single and Burst DMA channel [n] responds to single or burst requests. 1 Burst Only DMA channel [n] responds only to burst requests.

Writes

DMA Channel Useburst Set (DMAUSEBURSTSET)

Base 0x400F.F000
Offset 0x018
Type WO, reset 0x0000.0000



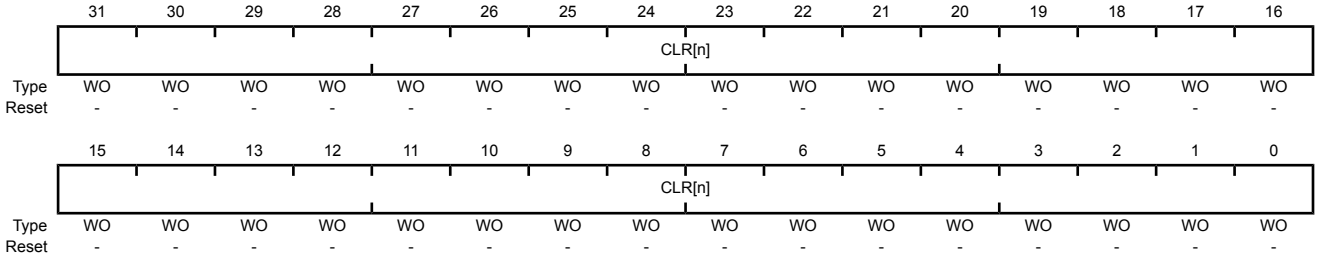
Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	W	0x00	Channel [n] Useburst Set Sets useburst bit on channel [n]. Value Description 0 No Effect Use the **DMAUSEBURSTCLR** register to clear bit [n] to 0. 1 Burst Only DMA channel [n] responds only to burst requests.

Register 11: DMA Channel Useburst Clear (DMAUSEBURSTCLR), offset 0x01C

Each bit of the **DMAUSEBURSTCLR** register represents the corresponding DMA channel. Writing a 1 enables `dma_sreq[n]` to generate requests.

DMA Channel Useburst Clear (DMAUSEBURSTCLR)

Base 0x400F.F000
Offset 0x01C
Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Useburst Clear Clears useburst bit on channel [n]. Value Description 0 No Effect Use the **DMAUSEBURSTSET** to set bit [n] to 1. 1 Single and Burst DMA channel [n] responds to single and burst requests.

Register 12: DMA Channel Request Mask Set (DMAREQMASKSET), offset 0x020

Each bit of the **DMAREQMASKSET** register represents the corresponding DMA channel. Writing a 1 disables DMA requests for the channel. Reading the register returns the request mask status. When a µDMA channel's request is masked, that means the peripheral can no longer request µDMA transfers. The channel can then be used for software-initiated transfers.

Reads

DMA Channel Request Mask Set (DMAREQMASKSET)

Base 0x400F.F000
Offset 0x020
Type RO, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R	0x00	Channel [n] Request Mask Status Returns the channel request mask status. Value Description 0 Enabled External requests are not masked for channel [n]. 1 Masked External requests are masked for channel [n].

Writes

DMA Channel Request Mask Set (DMAREQMASKSET)

Base 0x400F.F000
Offset 0x020
Type WO, reset 0x0000.0000

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

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

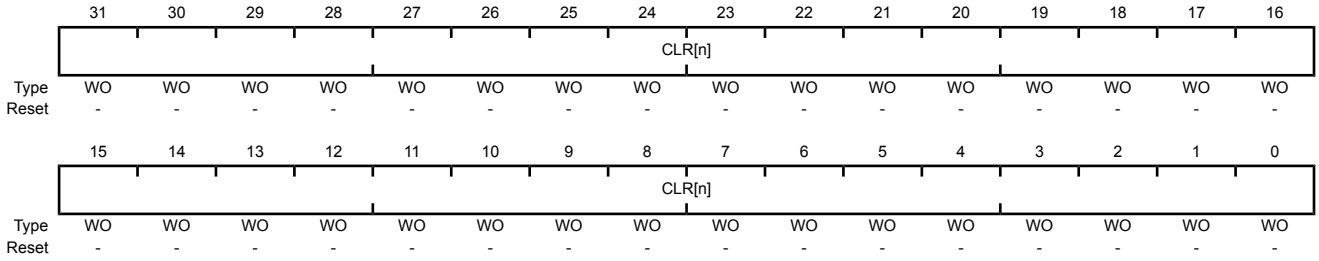
Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	W	0x00	Channel [n] Request Mask Set Masks (disables) the corresponding channel [n] from generating DMA requests. Value Description 0 No Effect Use the **DMAREQMASKCLR** register to clear the request mask. 1 Masked Masks (disables) DMA requests on channel [n].

Register 13: DMA Channel Request Mask Clear (DMAREQMASKCLR), offset 0x024

Each bit of the **DMAREQMASKCLR** register represents the corresponding DMA channel. Writing a 1 clears the request mask for the channel, and enables the channel to receive DMA requests.

DMA Channel Request Mask Clear (DMAREQMASKCLR)

Base 0x400F.F000
Offset 0x024
Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Request Mask Clear Set the appropriate bit to clear the DMA request mask for channel [n]. This will enable DMA requests for the channel. Value Description 0 No Effect Use the **DMAREQMASKSET** register to set the request mask. 1 Clear Mask Clears the request mask for the DMA channel. This enables DMA requests for the channel.

Register 14: DMA Channel Enable Set (DMAENASET), offset 0x028

Each bit of the **DMAENASET** register represents the corresponding DMA channel. Writing a 1 enables the DMA channel. Reading the register returns the enable status of the channels. If a channel is enabled but the request mask is set (**DMAREQMASKSET**), then the channel can be used for software-initiated transfers.

Reads

DMA Channel Enable Set (DMAENASET)

Base 0x400F.F000
Offset 0x028
Type RO, reset 0x0000.0000

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

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

Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R	0x00	Channel [n] Enable Status Returns the enable status of the channels. Value Description 0 Disabled 1 Enabled

Writes

DMA Channel Enable Set (DMAENASET)

Base 0x400F.F000
Offset 0x028
Type WO, reset 0x0000.0000

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

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

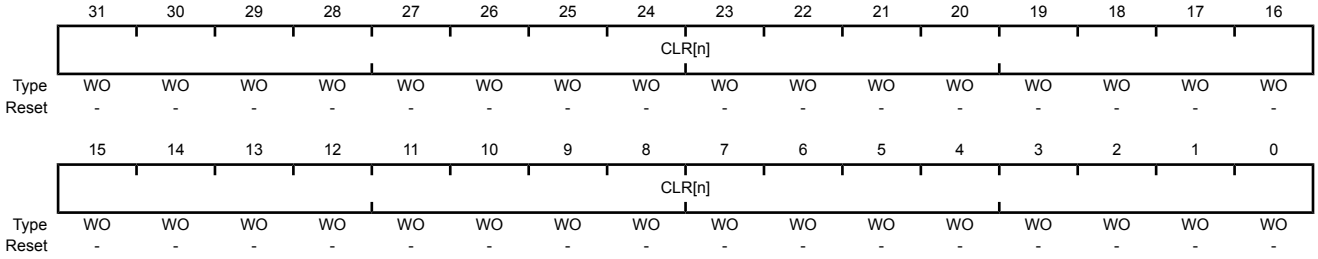
Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	W	0x00	Channel [n] Enable Set Enables the corresponding channels. **Note:** The controller disables a channel when it completes the DMA cycle. Value Description 0 No Effect Use the **DMAENACLR** register to disable a channel. 1 Enable Enables channel [n].

Register 15: DMA Channel Enable Clear (DMAENACLR), offset 0x02C

Each bit of the **DMAENACLR** register represents the corresponding DMA channel. Writing a 1 disables the specified DMA channel.

DMA Channel Enable Clear (DMAENACLR)

Base 0x400F.F000
Offset 0x02C
Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Clear Channel [n] Enable Set the appropriate bit to disable the corresponding DMA channel. **Note:** The controller disables a channel when it completes the DMA cycle. Value Description 0 No Effect Use the **DMAENASET** register to enable DMA channels. 1 Disable Disables channel [n].

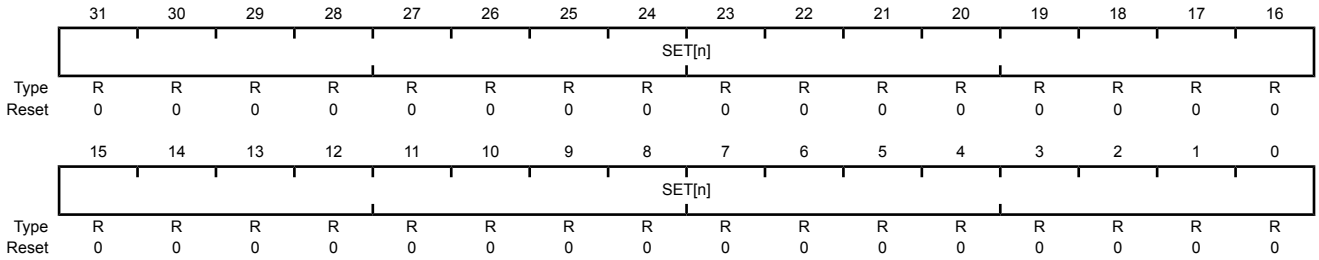
Register 16: DMA Channel Primary Alternate Set (DMAALTSET), offset 0x030

Each bit of the **DMAALTSET** register represents the corresponding DMA channel. Writing a 1 configures the DMA channel to use the alternate control data structure. Reading the register returns the status of which control data structure is in use for the corresponding DMA channel.

Reads

DMA Channel Primary Alternate Set (DMAALTSET)

Base 0x400F.F000
Offset 0x030
Type RO, reset 0x0000.0000

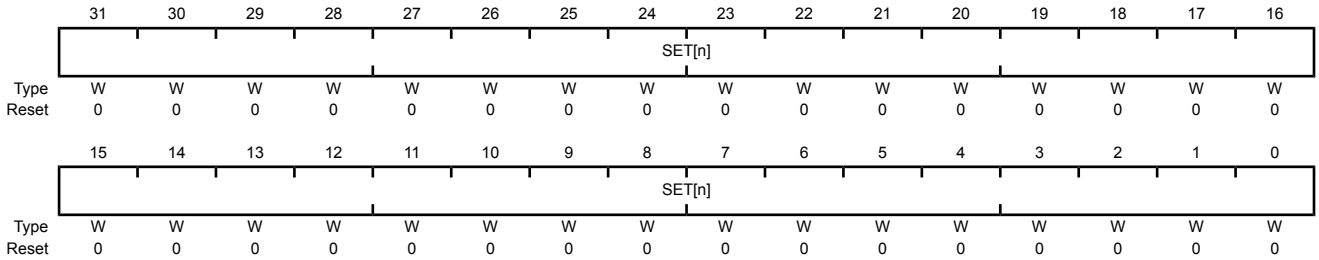


Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R	0x00	Channel [n] Alternate Status Returns the channel control data structure status. Value Description 0 Primary DMA channel [n] is using the primary control structure. 1 Alternate DMA channel [n] is using the alternate control structure.

Writes

DMA Channel Primary Alternate Set (DMAALTSET)

Base 0x400F.F000
Offset 0x030
Type WO, reset 0x0000.0000



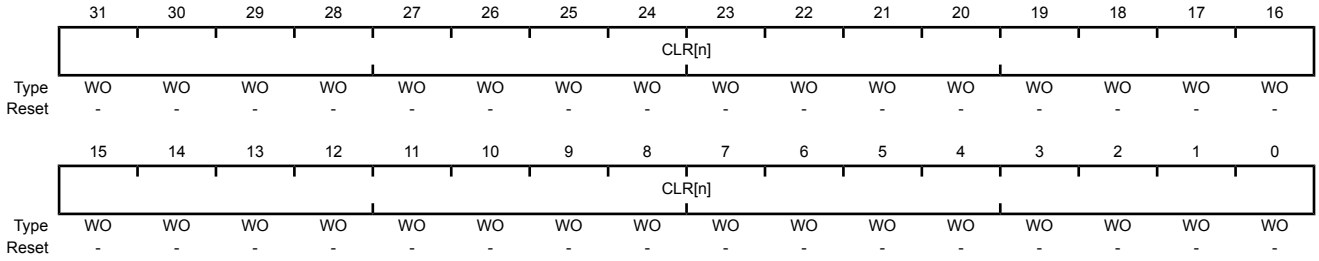
Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	W	0x00	Channel [n] Alternate Set Selects the alternate channel control data structure for the corresponding DMA channel. **Note:** For Ping-Pong and Scatter-Gather DMA cycle types, the controller automatically sets these bits to select the alternate channel control data structure. Value Description 0 No Effect Use the **DMAALTCLR** register to set bit [n] to 0. 1 Alternate Selects the alternate control data structure for channel [n].

Register 17: DMA Channel Primary Alternate Clear (DMAALTCLR), offset 0x034

Each bit of the **DMAALTCLR** register represents the corresponding DMA channel. Writing a 1 configures the DMA channel to use the primary control data structure.

DMA Channel Primary Alternate Clear (DMAALTCLR)

Base 0x400F.F000
Offset 0x034
Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Alternate Clear Set the appropriate bit to select the primary control data structure for the corresponding DMA channel. **Note:** For Ping-Pong and Scatter-Gather DMA cycle types, the controller sets these bits to select the primary channel control data structure. Value Description 0 No Effect Use the **DMAALTSET** register to select the alternate control data structure. 1 Primary Selects the primary control data structure for channel [n].

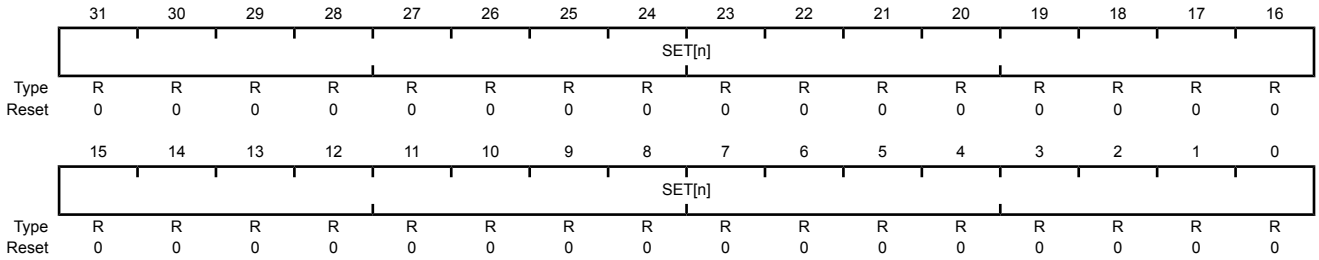
Register 18: DMA Channel Priority Set (DMAPRIOSET), offset 0x038

Each bit of the the **DMAPRIOSET** register represents the corresponding DMA channel. Writing a 1 configures the DMA channel to have a high priority level. Reading the register returns the status of the channel priority mask.

Reads

DMA Channel Priority Set (DMAPRIOSET)

Base 0x400F.F000
Offset 0x038
Type RO, reset 0x0000.0000

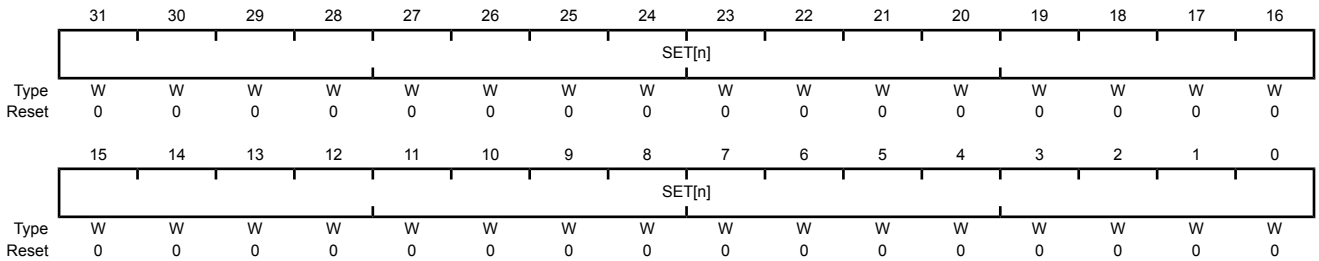


Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	R	0x00	Channel [n] Priority Status Returns the channel priority status. Value Description 0 Default Priority DMA channel [n] is using the default priority level. 1 High Priority DMA channel [n] is using a High Priority level.

Writes

DMA Channel Priority Set (DMAPRIOSET)

Base 0x400F.F000
Offset 0x038
Type WO, reset 0x0000.0000



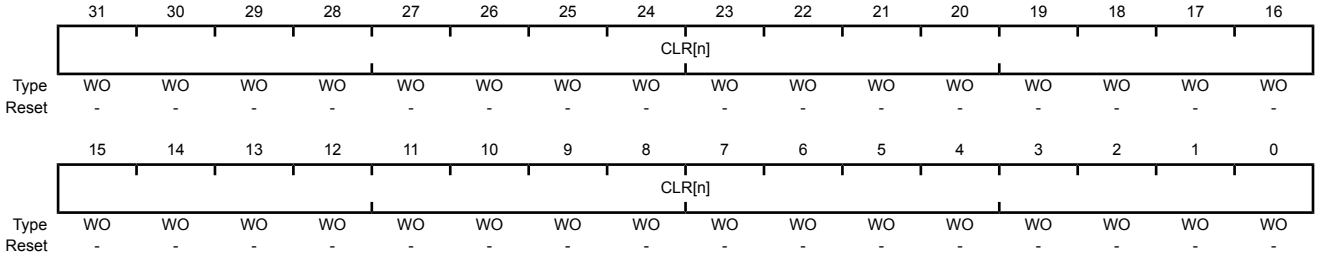
Bit/Field	Name	Type	Reset	Description
31:0	SET[n]	W	0x00	Channel [n] Priority Set Sets the channel priority to high. Value Description 0 No Effect Use the **DMAPRIOCLR** register to set channel [n] to the default priority level. 1 High Priority Sets DMA channel [n] to a High Priority level.

Register 19: DMA Channel Priority Clear (DMAPRIOCLR), offset 0x03C

Each bit of the **DMAPRIOCLR** register represents the corresponding DMA channel. Writing a 1 configures the DMA channel to have the default priority level.

DMA Channel Priority Clear (DMAPRIOCLR)

Base 0x400F.F000
Offset 0x03C
Type WO, reset -



Bit/Field	Name	Type	Reset	Description
31:0	CLR[n]	WO	-	Channel [n] Priority Clear Set the appropriate bit to clear the high priority level for the specified DMA channel. Value Description 0 No Effect Use the **DMAPRIOSET** register to set channel [n] to the High priority level. 1 Default Priority Sets DMA channel [n] to a Default priority level.

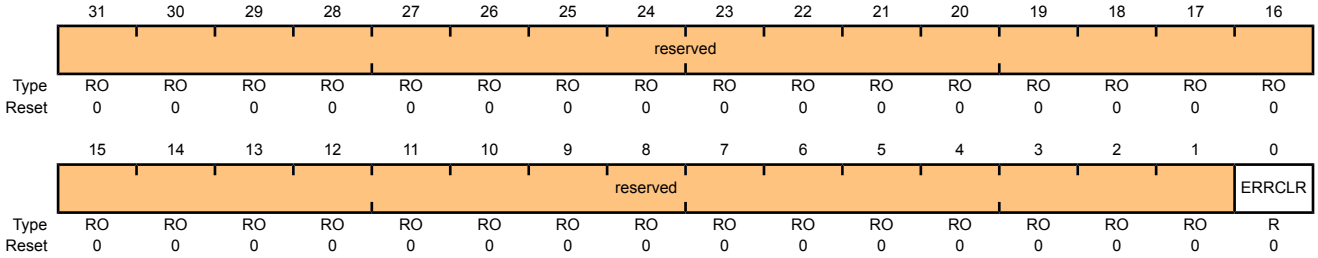
Register 20: DMA Bus Error Clear (DMAERRCLR), offset 0x04C

The **DMAERRCLR** register is used to read and clear the DMA bus error status. The error status will be set if the µDMA controller encountered a bus error while performing a DMA transfer. If a bus error occurs on a channel, that channel will be automatically disabled by the µDMA controller. The other channels are unaffected.

Reads

DMA Bus Error Clear (DMAERRCLR)

Base 0x400F.F000
Offset 0x04C
Type RO, reset 0x0000.0000

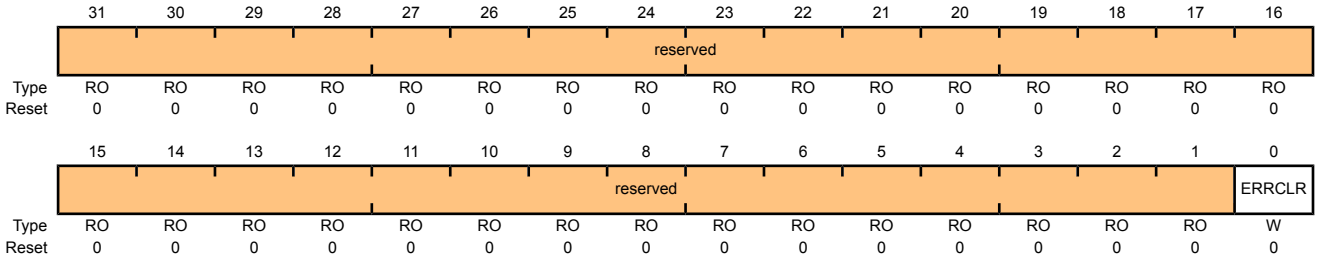


Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	ERRCLR	R	0	DMA Bus Error Status Value Description 0 Low No bus error is pending. 1 High Bus error is pending.

Writes

DMA Bus Error Clear (DMAERRCLR)

Base 0x400F.F000
Offset 0x04C
Type WO, reset 0x0000.0000



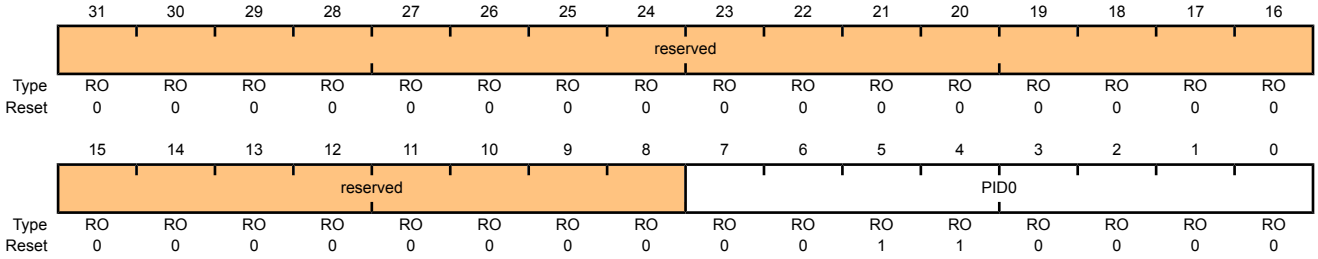
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	ERRCLR	W	0	DMA Bus Error Clear Clears the bus error. Value Description 0 No Effect Bus error status is unchanged. 1 Clear Clears a pending bus error.

Register 21: DMA Peripheral Identification 0 (DMAPeriphID0), offset 0xFE0

The **DMAPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

DMA Peripheral Identification 0 (DMAPeriphID0)

Base 0x400F.F000
Offset 0xFE0
Type RO, reset 0x0000.0030



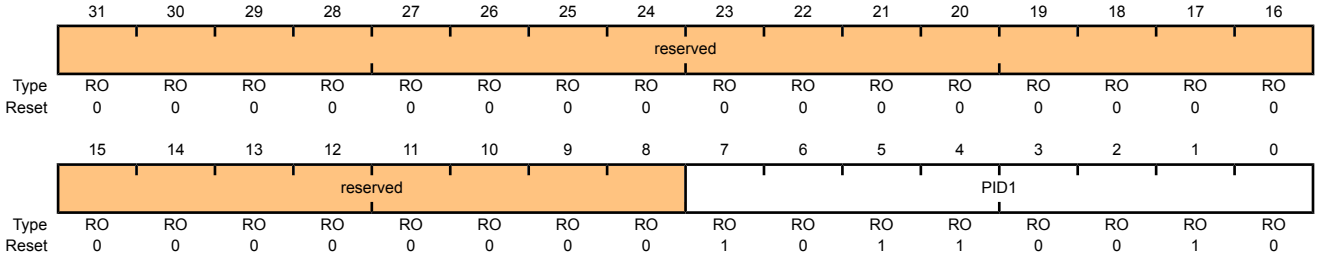
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x30	DMA Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 22: DMA Peripheral Identification 1 (DMAPeriphID1), offset 0xFE4

The **DMAPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

DMA Peripheral Identification 1 (DMAPeriphID1)

Base 0x400F.F000
Offset 0xFE4
Type RO, reset 0x0000.00B2



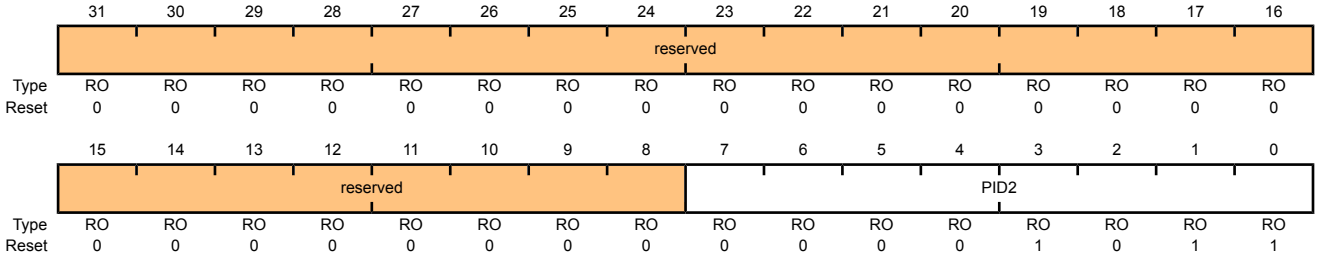
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0xB2	DMA Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

Register 23: DMA Peripheral Identification 2 (DMAPeriphID2), offset 0xFE8

The **DMAPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

DMA Peripheral Identification 2 (DMAPeriphID2)

Base 0x400F.F000
Offset 0xFE8
Type RO, reset 0x0000.000B



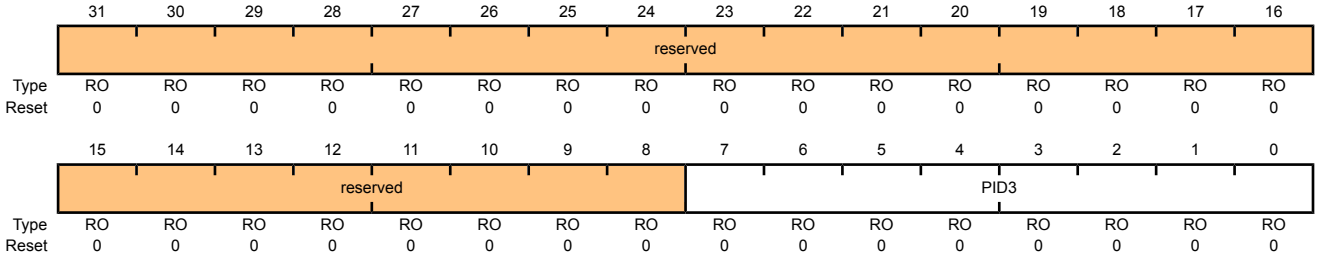
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x0B	DMA Peripheral ID Register[23:16] Can be used by software to identify the presence of this peripheral.

Register 24: DMA Peripheral Identification 3 (DMAPeriphID3), offset 0xFEC

The **DMAPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

DMA Peripheral Identification 3 (DMAPeriphID3)

Base 0x400F.F000
Offset 0xFEC
Type RO, reset 0x0000.0000



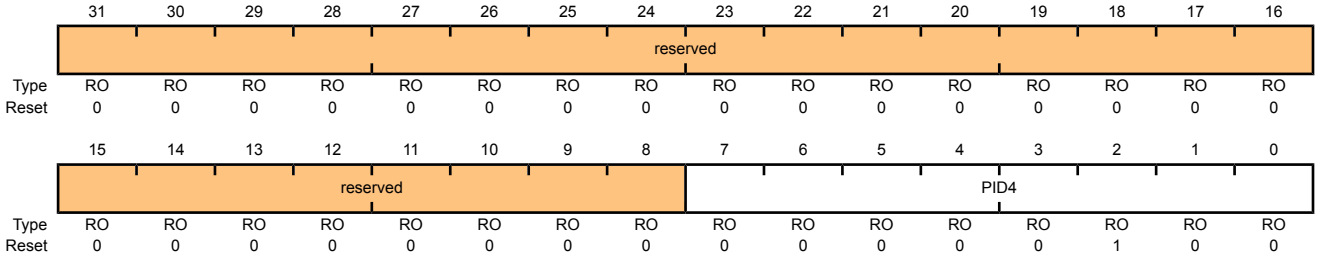
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x00	DMA Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

Register 25: DMA Peripheral Identification 4 (DMAPeriphID4), offset 0xFD0

The **DMAPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

DMA Peripheral Identification 4 (DMAPeriphID4)

Base 0x400F.F000
Offset 0xFD0
Type RO, reset 0x0000.0004



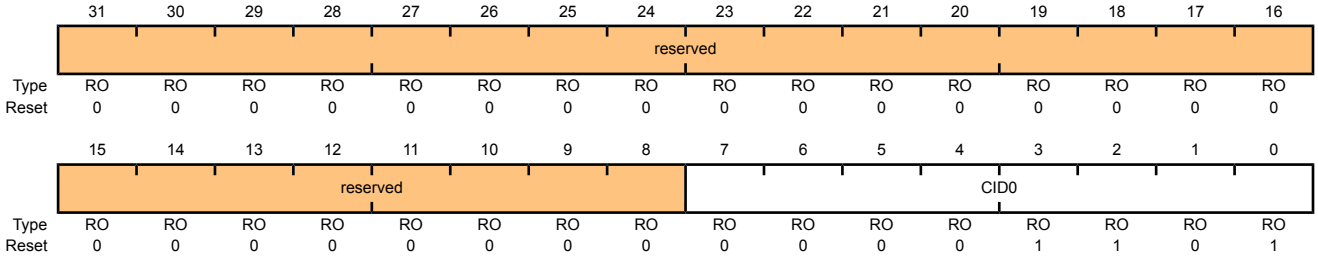
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x04	DMA Peripheral ID Register Can be used by software to identify the presence of this peripheral.

Register 26: DMA PrimeCell Identification 0 (DMAPCellID0), offset 0xFF0

The **DMAPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

DMA PrimeCell Identification 0 (DMAPCellID0)

Base 0x400F.F000
Offset 0xFF0
Type RO, reset 0x0000.000D



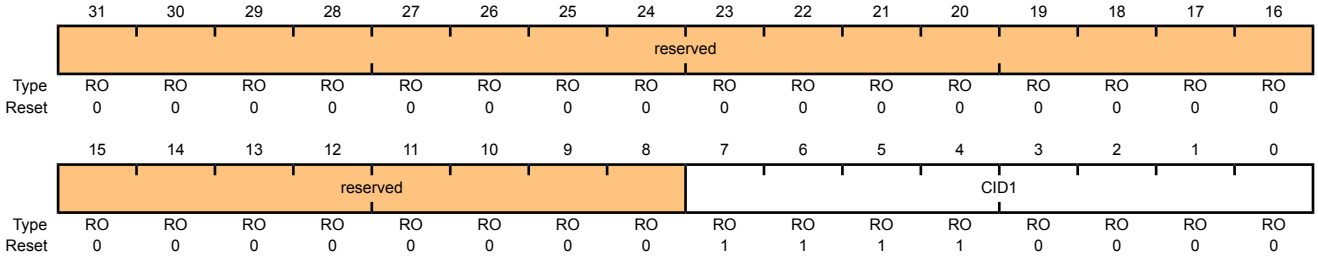
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	DMA PrimeCell ID Register[7:0] Provides software a standard cross-peripheral identification system.

Register 27: DMA PrimeCell Identification 1 (DMAPCellID1), offset 0xFF4

The **DMAPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

DMA PrimeCell Identification 1 (DMAPCellID1)

Base 0x400F.F000
Offset 0xFF4
Type RO, reset 0x0000.00F0



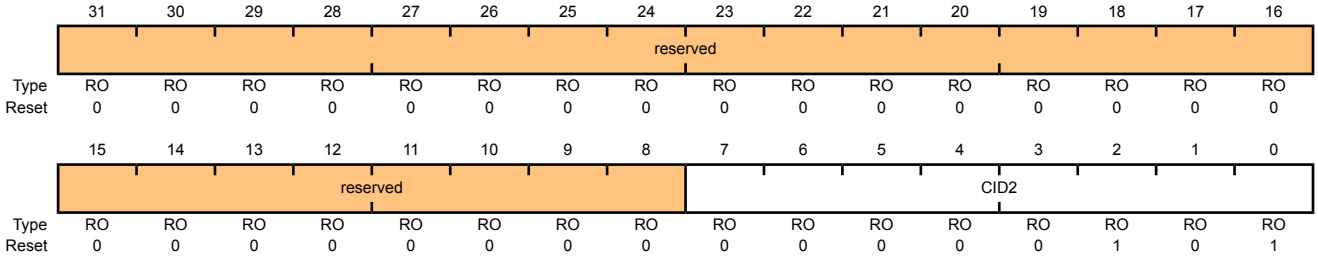
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	DMA PrimeCell ID Register[15:8] Provides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

Register 28: DMA PrimeCell Identification 2 (DMAPCellID2), offset 0xFF8

The **DMAPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

DMA PrimeCell Identification 2 (DMAPCellID2)

Base 0x400F.F000
Offset 0xFF8
Type RO, reset 0x0000.0005



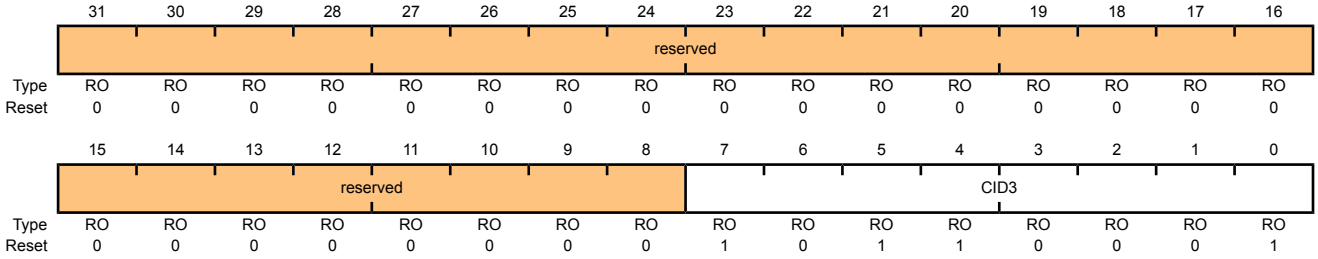
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	DMA PrimeCell ID Register[23:16] Provides software a standard cross-peripheral identification system.

Register 29: DMA PrimeCell Identification 3 (DMAPCellID3), offset 0xFFC

The **DMAPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

DMA PrimeCell Identification 3 (DMAPCellID3)

Base 0x400F.F000
Offset 0xFFC
Type RO, reset 0x0000.00B1



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	DMA PrimeCell ID Register[31:24] Provides software a standard cross-peripheral identification system.

10 General-Purpose Input/Outputs (GPIOs)

The GPIO module is composed of eight physical GPIO blocks, each corresponding to an individual GPIO port (Port A, Port B, Port C, Port D, Port E, Port F, Port G, Port H). The GPIO module supports 0-61 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

- 0-61 GPIOs, depending on configuration
- 5-V-tolerant input/outputs
- Two means of port access: either high speed (for single-cycle writes), or legacy for backwards-compatibility with existing code
- Programmable control for GPIO interrupts
  - Interrupt generation masking
  - Edge-triggered on rising, falling, or both
  - Level-sensitive on High or Low values
- Bit masking in both read and write operations through address lines
- Can initiate an ADC sample sequence
- Pins configured as digital inputs are Schmitt-triggered.
- Programmable control for GPIO pad configuration
  - Weak pull-up or pull-down resistors
  - 2-mA, 4-mA, and 8-mA pad drive for digital communication; up to four pads can be configured with an 18-mA pad drive for high-current applications
  - Slew rate control for the 8-mA drive
  - Open drain enables
  - Digital input enables

10.1 Functional Description

Important: All GPIO pins are tri-stated by default (**GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, and **GPIOPUR**=0), with the exception of the four JTAG/SWD pins (`PC[3:0]`). The JTAG/SWD pins default to their JTAG/SWD functionality (**GPIOAFSEL**=1, **GPIODEN=1** and **GPIOPUR**=1). A Power-On-Reset ($\overline{\text{POR}}$) or asserting $\overline{\text{RST}}$ 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 10-1 on page 256 and Figure 10-2 on page 257). The LM3S5749 microcontroller contains eight ports and thus eight of these physical GPIO blocks.

Figure 10-1. Digital I/O Pads

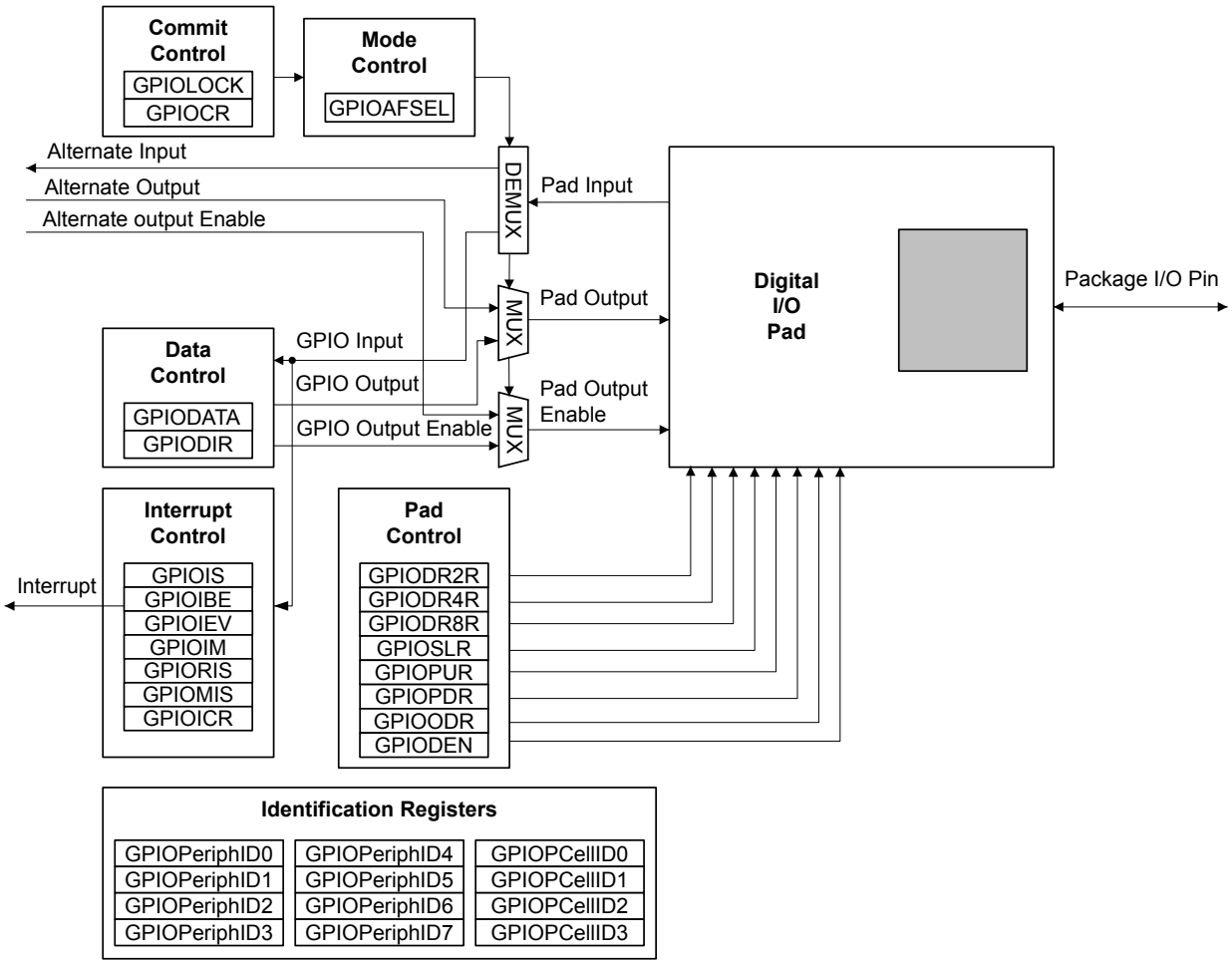
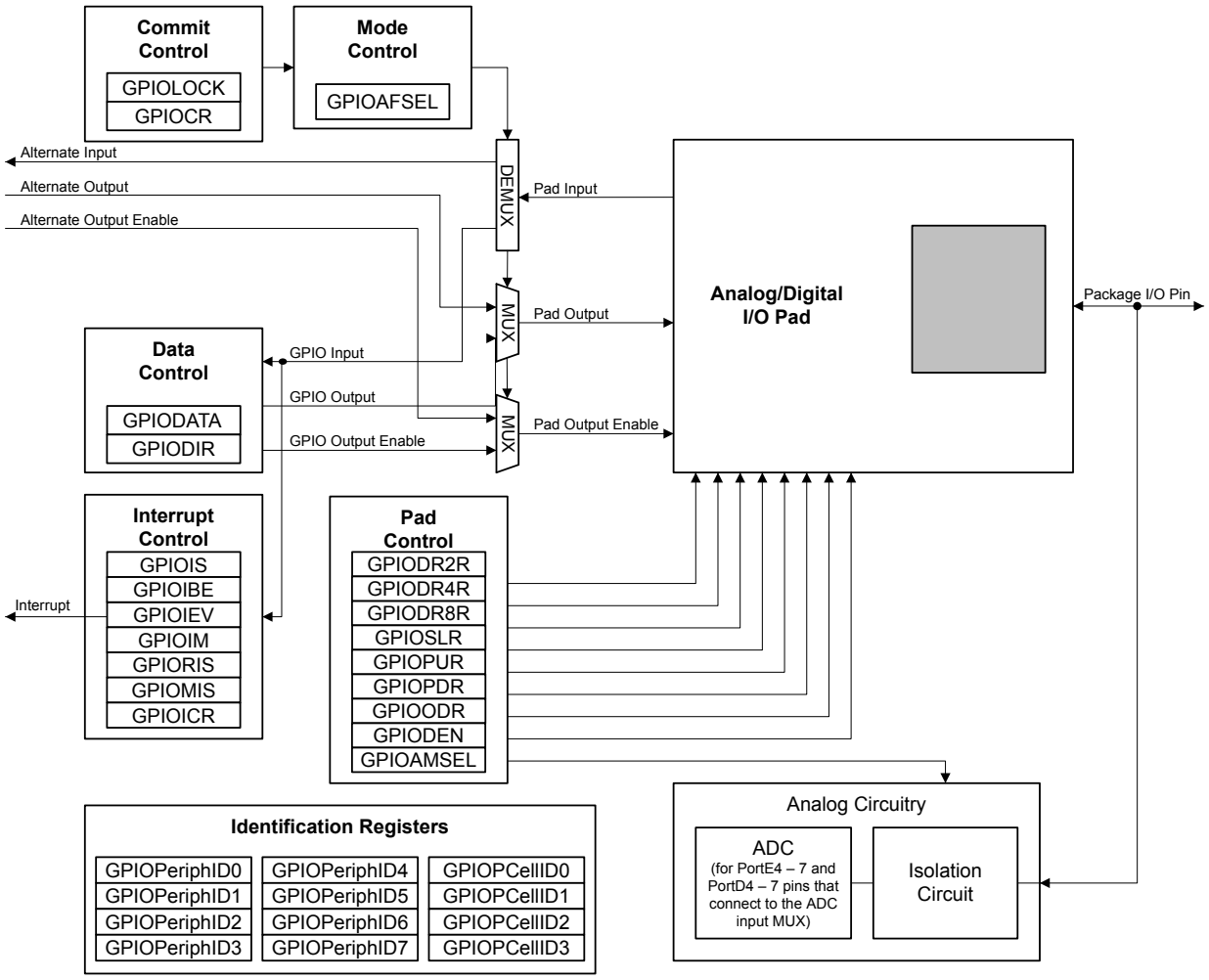


Figure 10-2. Analog/Digital I/O Pads



10.1.1 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.

10.1.1.1 Data Direction Operation

The **GPIO Direction (GPIODIR)** register (see page 265) 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.

10.1.1.2 Data Register Operation

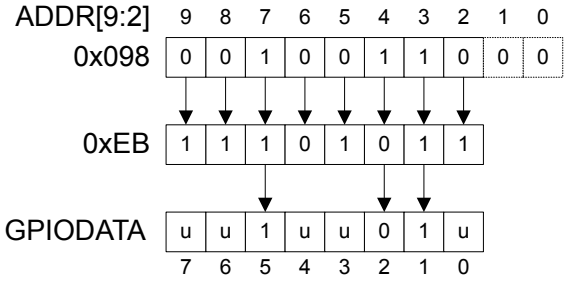
To aid in the efficiency of software, the GPIO ports allow for the modification of individual bits in the **GPIO Data (GPIODATA)** register (see page 264) by using bits [9:2] of the address bus as a mask. This allows software drivers to modify individual GPIO pins in a single instruction, without affecting 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 10-3 on page 258, where u is data unchanged by the write.

Figure 10-3. 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 10-4 on page 258.

Figure 10-4. GPIODATA Read Example

10.1.2 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 266)
- **GPIO Interrupt Both Edges (GPIOIBE)** register (see page 267)
- **GPIO Interrupt Event (GPIOIEV)** register (see page 268)

Interrupts are enabled/disabled via the **GPIO Interrupt Mask (GPIOIM)** register (see page 269).

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 270 and page 271). As the name implies, the **GPIOMIS** register only shows interrupt conditions that are allowed to be passed to the controller. The **GPIORIS** register indicates that a GPIO pin meets the conditions for an interrupt, but has not necessarily been sent to the controller.

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

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

Interrupts are cleared by writing a 1 to the appropriate bit of the **GPIO Interrupt Clear (GPIOICR)** register (see page 273).

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.

10.1.3 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 274), 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.

Note: If any pin is to be used as an ADC input, the appropriate bit in **GPIOAMSEL** must be written to 1 to disable the analog isolation circuit.

10.1.4 Commit Control

The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the NMI pin (PB7) and the four JTAG/SWD pins (PC[3:0]). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 274), **GPIO Pull-Up Select (GPIOPUR)** register (see page 280), and **GPIO Digital Enable (GPIODEN)** register (see page 284) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 286) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 287) have been set to 1.

10.1.5 Pad Control

The pad control registers allow for GPIO pad configuration by software based on the application requirements. The pad control registers include the **GPIODR2R**, **GPIODR4R**, **GPIODR8R**, **GPIOODR**, **GPIOPUR**, **GPIOPDR**, **GPIOSLR**, and **GPIODEN** registers. These registers control drive strength, open-drain configuration, pull-up and pull-down resistors, slew-rate control and digital input enable.

For special high-current applications, the GPIO output buffers may be used with the following restrictions. With the GPIO pins configured as 8-mA output drivers, a total of four GPIO outputs may be used to sink current loads up to 18 mA each. At 18-mA sink current loading, the V_{OL} value is specified as 1.2 V. The high-current GPIO package pins must be selected such that there are only

a maximum of two per side of the physical package with the total number of high-current GPIO outputs not exceeding four for the entire package.

10.1.6 Identification

The identification registers configured at reset allow software to detect and identify the module as a GPIO block. The identification registers include the **GPIOPeriphID0**-**GPIOPeriphID7** registers as well as the **GPIOPCellID0**-**GPIOPCellID3** registers.

10.2 Initialization and Configuration

The GPIO modules may be accessed via two different memory apertures. The legacy aperture is backwards-compatible with previous Stellaris parts and offers two-cycle access time to all GPIO registers. The high-speed aperture offers the same register map but provides single-cycle access times. These apertures are mutually exclusive. The aperture enabled for a given GPIO port is controlled by the appropriate bit in the **GPIOHSCTL** register (see page 97).

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 four JTAG pins) are configured out of reset to be undriven (tristate): **GPIOAFSEL**=0, **GPIODEN**=0, **GPIOPDR**=0, and **GPIOPUR**=0. Table 10-1 on page 260 shows all possible configurations of the GPIO pads and the control register settings required to achieve them. Table 10-2 on page 261 shows how a rising edge interrupt would be configured for pin 2 of a GPIO port.

Table 10-1. 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 Input (QEI)	1	X	0	1	?	?	X	X	X	X
Digital Output (PWM)	1	X	0	1	?	?	?	?	?	?
Digital Output (Timer PWM)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (SSI)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (UART)	1	X	0	1	?	?	?	?	?	?
Analog Input (Comparator)	0	0	0	0	0	0	X	X	X	X
Digital Output (Comparator)	1	X	0	1	?	?	?	?	?	?

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

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

Table 10-2. 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)

10.3 Register Map

Table 10-3 on page 262 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 (legacy): 0x4000.4000
- GPIO Port A (high-speed): 0x4005.8000
- GPIO Port B (legacy): 0x4000.5000
- GPIO Port B (high-speed): 0x4005.9000
- GPIO Port C (legacy): 0x4000.6000
- GPIO Port C (high-speed): 0x4005.A000
- GPIO Port D (legacy): 0x4000.7000
- GPIO Port D (high-speed): 0x4005.B000
- GPIO Port E (legacy): 0x4002.4000
- GPIO Port E (high-speed): 0x4005.C000
- GPIO Port F (legacy): 0x4002.5000
- GPIO Port F (high-speed): 0x4005.D000
- GPIO Port G (legacy): 0x4002.6000
- GPIO Port G (high-speed): 0x4005.E000
- GPIO Port H (legacy): 0x4002.7000
- GPIO Port H (high-speed): 0x4005.F000

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

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

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

The default reset value for the **GPIOCR** register is 0x0000.00FF for all GPIO pins, with the exception of the NMI pin and the four JTAG/SWD pins (PB7 and PC[3:0]). To ensure that the JTAG port is not accidentally programmed as a GPIO, these four pins default to non-committable. To ensure that the NMI pin is not accidentally programmed as the non-maskable interrupt pin, it defaults to non-committable. Because of this, the default reset value of **GPIOCR** for GPIO Port B is 0x0000.007F while the default reset value of GPIOCR for Port C is 0x0000.00F0.

Table 10-3. GPIO Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPIODATA	R/W	0x0000.0000	GPIO Data	264
0x400	GPIODIR	R/W	0x0000.0000	GPIO Direction	265
0x404	GPIOIS	R/W	0x0000.0000	GPIO Interrupt Sense	266
0x408	GPIOIBE	R/W	0x0000.0000	GPIO Interrupt Both Edges	267
0x40C	GPIOIEV	R/W	0x0000.0000	GPIO Interrupt Event	268
0x410	GPIOIM	R/W	0x0000.0000	GPIO Interrupt Mask	269
0x414	GPIORIS	RO	0x0000.0000	GPIO Raw Interrupt Status	270
0x418	GPIOMIS	RO	0x0000.0000	GPIO Masked Interrupt Status	271
0x41C	GPIOICR	W1C	0x0000.0000	GPIO Interrupt Clear	273
0x420	GPIOAFSEL	R/W	-	GPIO Alternate Function Select	274
0x500	GPIODR2R	R/W	0x0000.00FF	GPIO 2-mA Drive Select	276
0x504	GPIODR4R	R/W	0x0000.0000	GPIO 4-mA Drive Select	277
0x508	GPIODR8R	R/W	0x0000.0000	GPIO 8-mA Drive Select	278
0x50C	GPIOODR	R/W	0x0000.0000	GPIO Open Drain Select	279
0x510	GPIOPUR	R/W	-	GPIO Pull-Up Select	280
0x514	GPIOPDR	R/W	0x0000.0000	GPIO Pull-Down Select	282
0x518	GPIOSLR	R/W	0x0000.0000	GPIO Slew Rate Control Select	283
0x51C	GPIODEN	R/W	-	GPIO Digital Enable	284
0x520	GPIOLOCK	R/W	0x0000.0001	GPIO Lock	286
0x524	GPIOCR	-	-	GPIO Commit	287
0x528	GPIOAMSEL	R/W	0x0000.0000	GPIO Analog Mode Select	289

Offset	Name	Type	Reset	Description	See page
0xFD0	GPIOPeriphID4	RO	0x0000.0000	GPIO Peripheral Identification 4	291
0xFD4	GPIOPeriphID5	RO	0x0000.0000	GPIO Peripheral Identification 5	292
0xFD8	GPIOPeriphID6	RO	0x0000.0000	GPIO Peripheral Identification 6	293
0xFDC	GPIOPeriphID7	RO	0x0000.0000	GPIO Peripheral Identification 7	294
0xFE0	GPIOPeriphID0	RO	0x0000.0061	GPIO Peripheral Identification 0	295
0xFE4	GPIOPeriphID1	RO	0x0000.0000	GPIO Peripheral Identification 1	296
0xFE8	GPIOPeriphID2	RO	0x0000.0018	GPIO Peripheral Identification 2	297
0xFEC	GPIOPeriphID3	RO	0x0000.0001	GPIO Peripheral Identification 3	298
0xFF0	GPIOPCellID0	RO	0x0000.000D	GPIO PrimeCell Identification 0	299
0xFF4	GPIOPCellID1	RO	0x0000.00F0	GPIO PrimeCell Identification 1	300
0xFF8	GPIOPCellID2	RO	0x0000.0005	GPIO PrimeCell Identification 2	301
0xFFC	GPIOPCellID3	RO	0x0000.00B1	GPIO PrimeCell Identification 3	302

10.4 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 265).

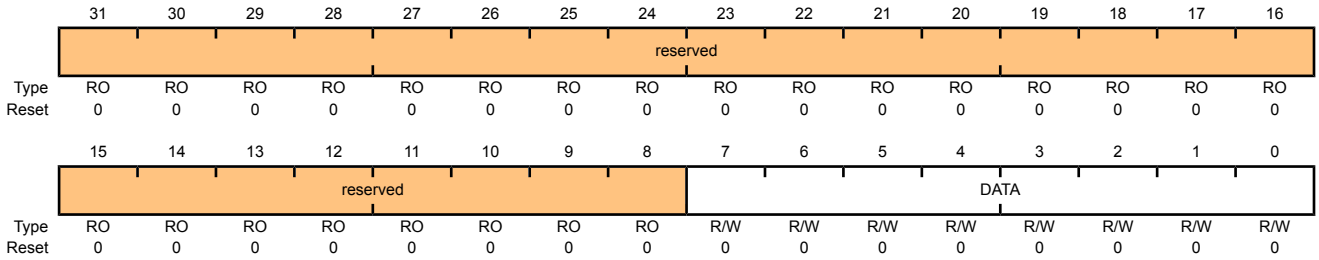
In order to write to **GPIODATA**, the corresponding bits in the mask, resulting from the address bus bits [9:2], must be High. Otherwise, the bit values remain unchanged by the write.

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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x000
Type R/W, reset 0x0000.0000



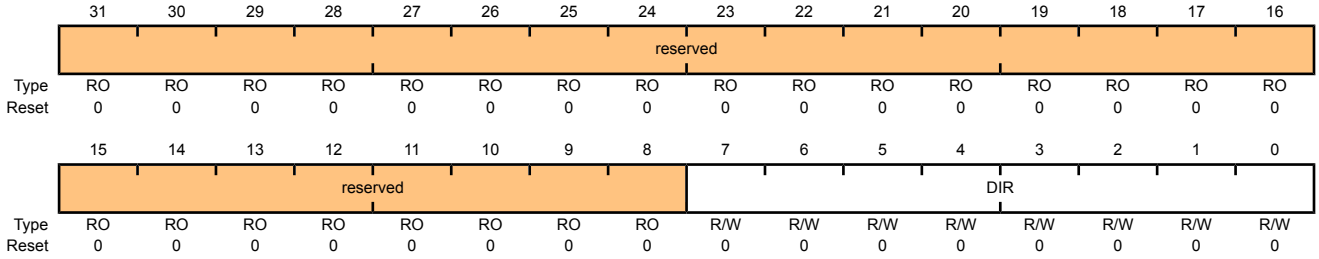
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x00	GPIO Data This register is virtually 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 257 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x400
Type R/W, reset 0x0000.0000



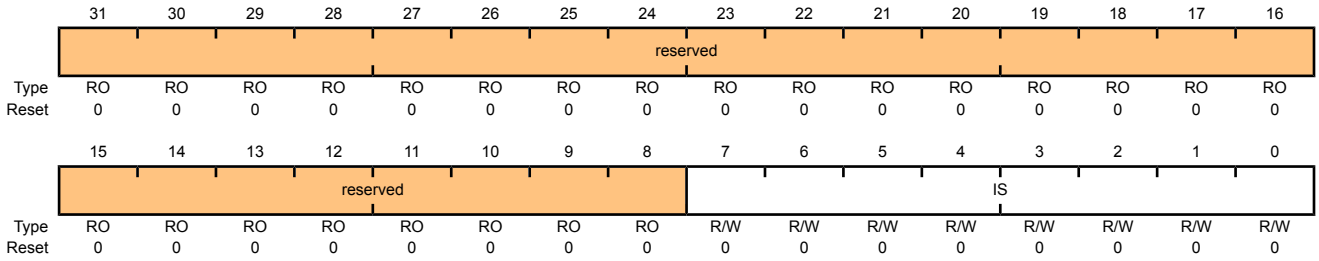
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DIR	R/W	0x00	GPIO Data Direction The `DIR` values 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x404
Type R/W, reset 0x0000.0000



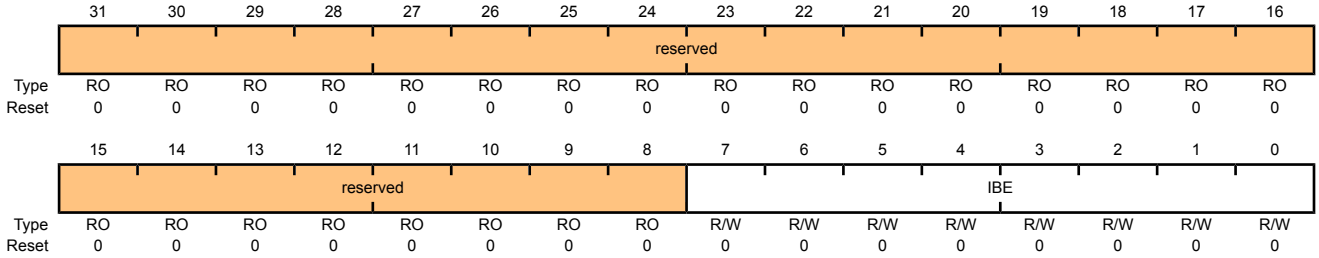
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IS	R/W	0x00	GPIO Interrupt Sense The `IS` values 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 266) 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 268). 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x408
Type R/W, reset 0x0000.0000



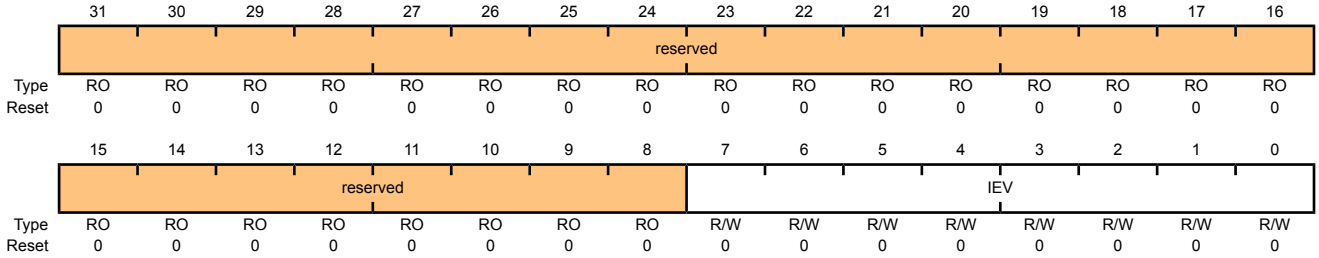
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IBE	R/W	0x00	GPIO Interrupt Both Edges The IBE values are defined as follows: Value Description 0 Interrupt generation is controlled by the **GPIO Interrupt Event (GPIOIEV)** register (see page 268). 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 266). 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x40C
Type R/W, reset 0x0000.0000



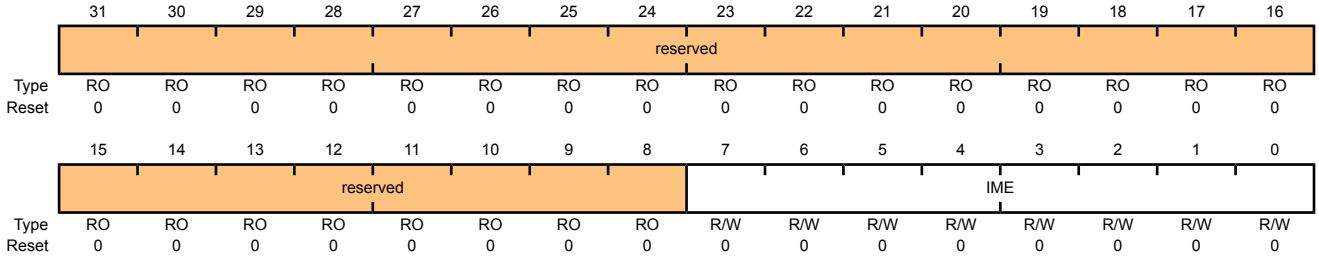
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IEV	R/W	0x00	GPIO Interrupt Event The `IEV` values 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x410
Type R/W, reset 0x0000.0000



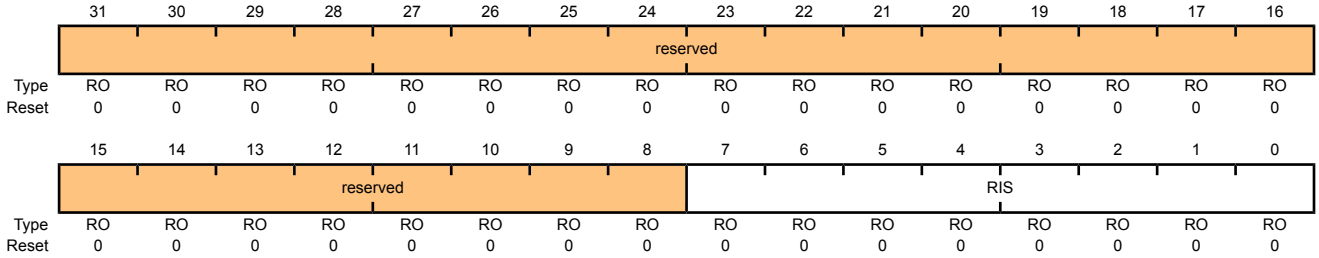
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	IME	R/W	0x00	GPIO Interrupt Mask Enable The `IME` 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 269). 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x414
Type RO, reset 0x0000.0000



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

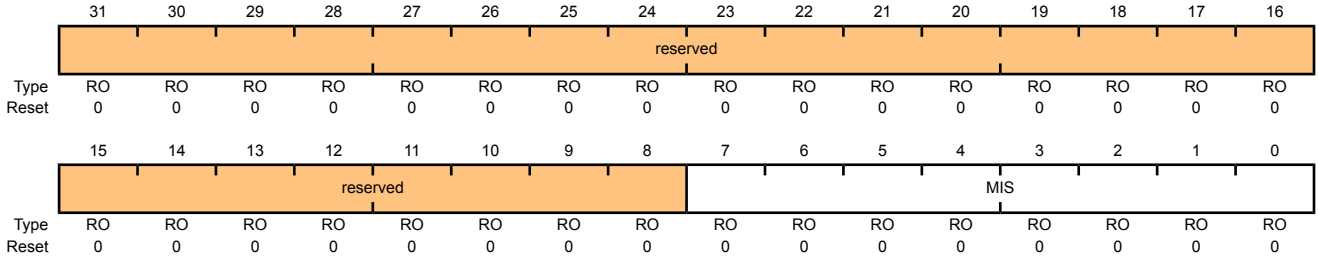
In addition to providing GPIO functionality, PB4 can also be used as an external trigger for the ADC. If PB4 is configured as a non-masked interrupt pin (the appropriate bit of GPIOIM is set to 1), not only is an interrupt for PortB generated, but an external trigger signal is sent to the ADC. If the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

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

GPIOMIS is the state of the interrupt after masking.

GPIO Masked Interrupt Status (GPIOMIS)

GPIO Port A (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x418
Type RO, reset 0x0000.0000



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

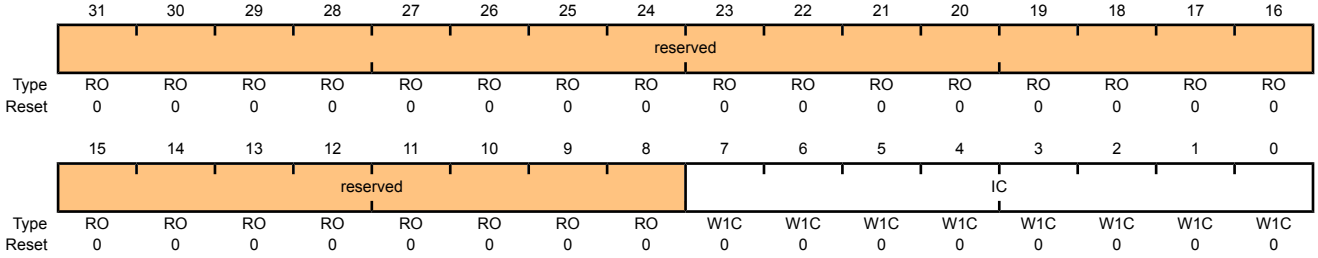
Bit/Field	Name	Type	Reset	Description
7:0	MIS	RO	0x00	GPIO Masked Interrupt Status Masked value of interrupt due to corresponding pin. The MIS values are defined as follows: Value Description 0 Corresponding GPIO line interrupt not active. 1 Corresponding GPIO line asserting interrupt.

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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x41C
Type W1C, reset 0x0000.0000



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

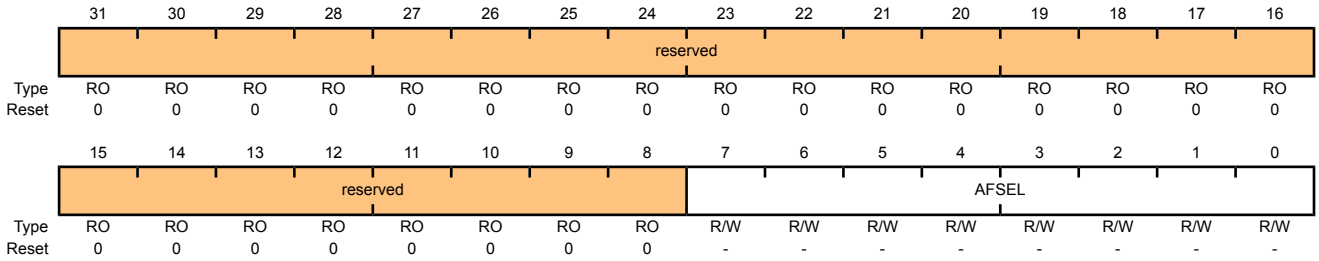
The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the `NMI` pin (`PB7`) and the four JTAG/SWD pins (`PC[3:0]`). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 274), **GPIO Pull-Up Select (GPIOPUR)** register (see page 280), and **GPIO Digital Enable (GPIODEN)** register (see page 284) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 286) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 287) have been set to 1.

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

Caution – It is possible to create a software sequence that prevents the debugger from connecting to the Stellaris® microcontroller. If the program code loaded into flash immediately changes the JTAG pins to their GPIO functionality, the debugger may not have enough time to connect and halt the controller before the JTAG pin functionality switches. 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x420
Type R/W, reset -



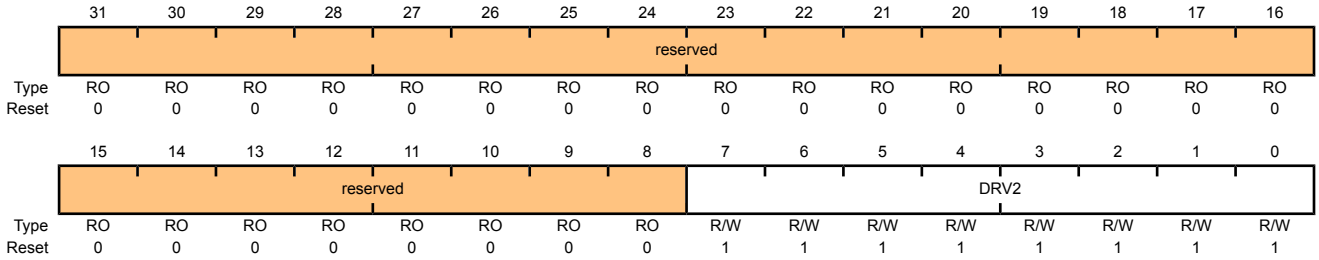
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	AFSEL	R/W	-	GPIO Alternate Function Select The 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**, **GPIOPUR**, and **GPIODEN** registers are 0x0000.0000 for all GPIO pins, with the exception of the four JTAG/SWD pins (PC[3:0]). These four pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x500
Type R/W, reset 0x0000.00FF



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 if accessing GPIO via the legacy memory aperture. If using high-speed access, the change is effective on the next clock cycle.

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

The **GPIODR4R** register is the 4-mA drive control register. 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
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 if accessing GPIO via the legacy memory aperture. If using high-speed access, the change is effective on the next clock cycle.

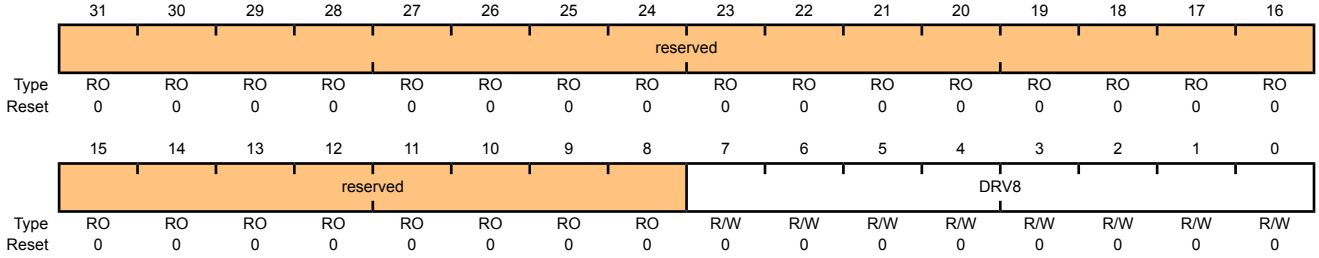
Register 13: GPIO 8-mA Drive Select (GPIODR8R), offset 0x508

The **GPIODR8R** register is the 8-mA drive control register. 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. The 8-mA setting is also used for high-current operation.

Note: There is no configuration difference between 8-mA and high-current operation. The additional current capacity results from a shift in the V_{OH}/V_{OL} levels. See "Recommended DC Operating Conditions" on page 749 for further information.

GPIO 8-mA Drive Select (GPIODR8R)

GPIO Port A (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x508
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	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 if accessing GPIO via the legacy memory aperture. If using high-speed access, the change is effective on the next clock cycle.

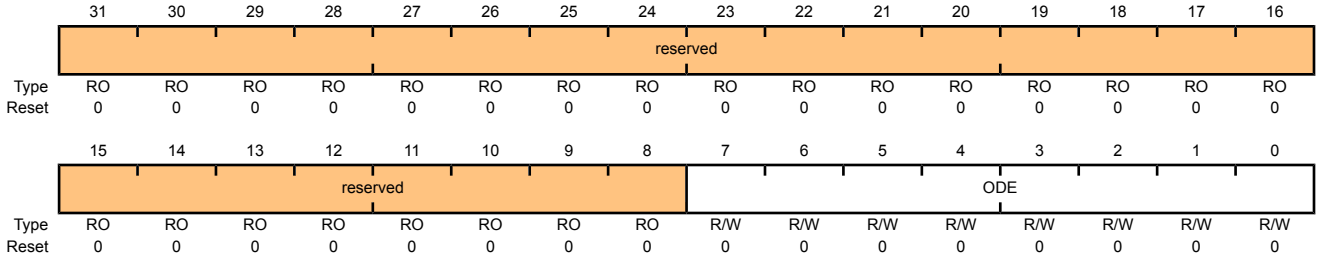
Register 14: GPIO Open Drain Select (GPIOODR), offset 0x50C

The **GPIOODR** register is the open drain control register. Setting a bit in this register enables the open drain configuration of the corresponding GPIO pad. When open drain mode is enabled, the corresponding bit should also be set in the **GPIO Digital Input Enable (GPIODEN)** register (see page 284). 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^2C module, in addition to configuring the pin to open drain, the **GPIO Alternate Function Select (GPIOAFSEL)** register bits for the I^2C clock and data pins should be set to 1 (see examples in “Initialization and Configuration” on page 260).

GPIO Open Drain Select (GPIOODR)

GPIO Port A (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x50C
Type R/W, reset 0x0000.0000



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

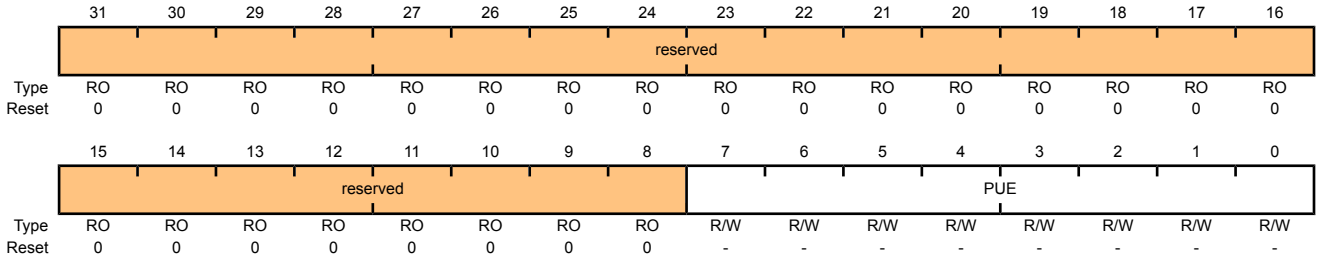
Register 15: GPIO Pull-Up Select (GPIOPUR), offset 0x510

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 282). Write access to this register is protected with the **GPIOCR** register. Bits in **GPIOCR** that are set to 0 will prevent writes to the equivalent bit in this register.

Note: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the `NMI` pin (`PB7`) and the four JTAG/SWD pins (`PC[3:0]`). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 274), **GPIO Pull-Up Select (GPIOPUR)** register (see page 280), and **GPIO Digital Enable (GPIODEN)** register (see page 284) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 286) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 287) have been set to 1.

GPIO Pull-Up Select (GPIOPUR)

GPIO Port A (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x510
Type R/W, reset -



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.

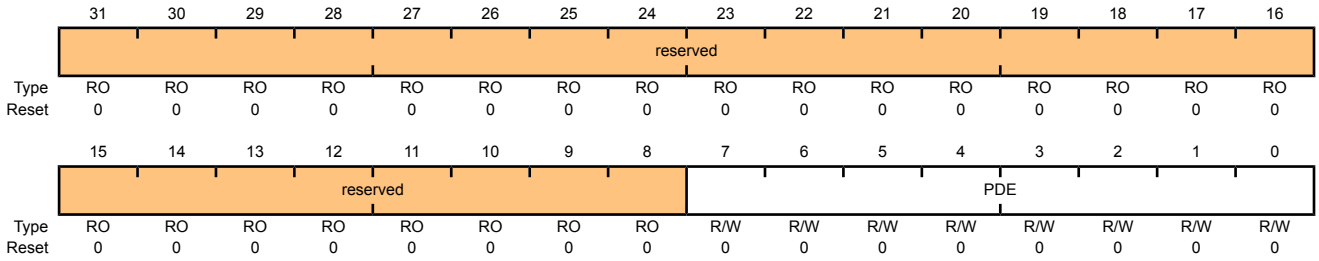
Bit/Field	Name	Type	Reset	Description
7:0	PUE	R/W	-	Pad Weak Pull-Up Enable A write of 1 to **GPIOPDR[n]** clears the corresponding **GPIOPUR[n]** enables. The change is effective on the second clock cycle after the write. **Note:** The default reset value for the **GPIOAFSEL**, **GPIOPUR**, and **GPIODEN** registers are 0x0000.0000 for all GPIO pins, with the exception of the four JTAG/SWD pins (`PC[3:0]`). These four pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers for Port C is 0x0000.000F.

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

The **GPIOPDR** register is the pull-down control register. When a bit is set to 1, it enables a weak pull-down resistor on the corresponding GPIO signal. Setting a bit in **GPIOPDR** automatically clears the corresponding bit in the **GPIO Pull-Up Select (GPIOPUR)** register (see page 280).

GPIO Pull-Down Select (GPIOPDR)

GPIO Port A (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x514
Type R/W, reset 0x0000.0000



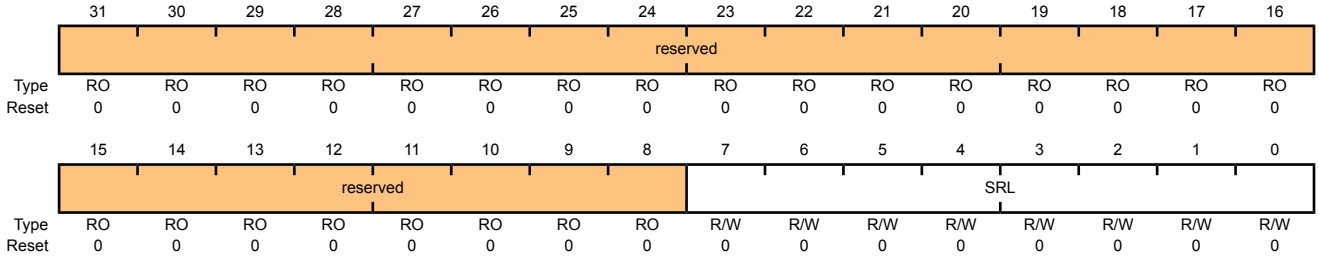
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PDE	R/W	0x00	Pad Weak Pull-Down Enable A write 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 278).

GPIO Slew Rate Control Select (GPIOSLR)

GPIO Port A (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x518
Type R/W, reset 0x0000.0000



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

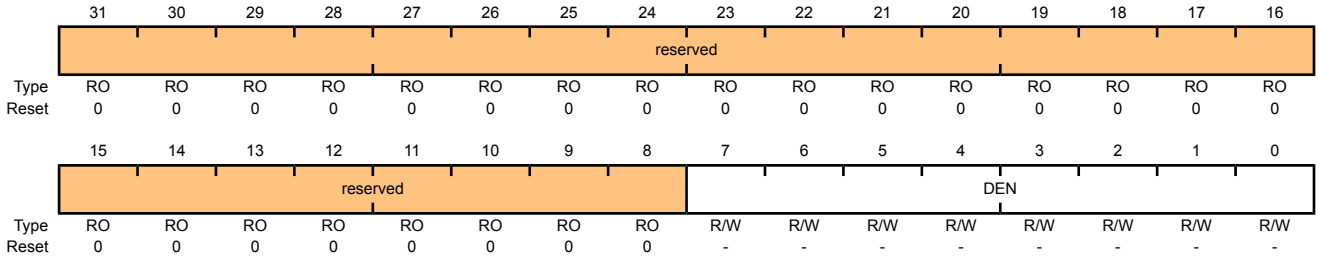
Note: Pins configured as digital inputs are Schmitt-triggered.

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

Note: The GPIO commit control registers provide a layer of protection against accidental programming of critical hardware peripherals. Protection is currently provided for the NMI pin (PB7) and the four JTAG/SWD pins (PC[3:0]). Writes to protected bits of the **GPIO Alternate Function Select (GPIOAFSEL)** register (see page 274), **GPIO Pull-Up Select (GPIOPUR)** register (see page 280), and **GPIO Digital Enable (GPIODEN)** register (see page 284) are not committed to storage unless the **GPIO Lock (GPIOLOCK)** register (see page 286) has been unlocked and the appropriate bits of the **GPIO Commit (GPIOCR)** register (see page 287) have been set to 1.

GPIO Digital Enable (GPIODEN)

GPIO Port A (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x51C
Type R/W, reset -



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	DEN	R/W	-	Digital Enable The DEN values are defined as follows: Value Description 0 Digital functions disabled. 1 Digital functions enabled. **Note:** The default reset value for the **GPIOAFSEL**, **GPIOPUR**, and **GPIODEN** registers are 0x0000.0000 for all GPIO pins, with the exception of the four JTAG/SWD pins (PC[3:0]). These four pins default to JTAG/SWD functionality. Because of this, the default reset value of these registers for Port C is 0x0000.000F.

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

The **GPIOLOCK** register enables write access to the **GPIOCR** register (see page 287). Writing 0x0x4C4F.434B to the **GPIOLOCK** register will unlock the **GPIOCR** register. Writing any other value to the **GPIOLOCK** register re-enables the locked state. Reading the **GPIOLOCK** register returns the lock status rather than the 32-bit value that was previously written. Therefore, when write accesses are disabled, or locked, reading the **GPIOLOCK** register returns 0x00000001. When write accesses are enabled, or unlocked, reading the **GPIOLOCK** register returns 0x00000000.

GPIO Lock (GPIOLOCK)

GPIO Port A (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x520
Type R/W, reset 0x0000.0001

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

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

Bit/Field	Name	Type	Reset	Description
31:0	LOCK	R/W	0x0000.0001	GPIO Lock

A write of the value 0x4C4F.434B unlocks the **GPIO Commit (GPIOCR)** register for write access.

A write of any other value or a write to the **GPIOCR** register reapplies the lock, preventing any register updates. A read of this register returns the following values:

Value	Description
0x0000.0001	locked
0x0000.0000	unlocked

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

The **GPIOCR** register is the commit register. The value of the **GPIOCR** register determines which bits of the **GPIOAFSEL**, **GPIOPUR**, and **GPIODEN** registers are committed when a write to these registers is performed. If a bit in the **GPIOCR** register is zero, the data being written to the corresponding bit in the **GPIOAFSEL**, **GPIOPUR**, or **GPIODEN** registers cannot be committed and retains its previous value. If a bit in the **GPIOCR** register is set, the data being written to the corresponding bit of the **GPIOAFSEL**, **GPIOPUR**, or **GPIODEN** registers is committed to the register and reflects the new value.

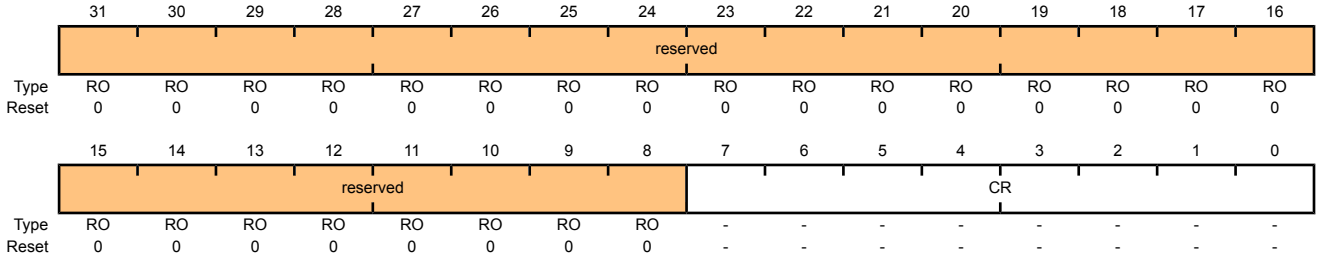
The contents of the **GPIOCR** register can only be modified if the **GPIOLOCK** register is unlocked. Writes to the **GPIOCR** register are ignored if the **GPIOLOCK** register is locked.

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

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

GPIO Commit (GPIOCR)

GPIO Port A (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x524
Type -, reset -



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	CR	-	-	GPIO Commit On a bit-wise basis, any bit set allows the corresponding **GPIOAFSEL**, **GPIOPUR**, or **GPIODEN** registers to be written. **Note:** The default register type for the **GPIOCR** register is RO for all GPIO pins with the exception of the `NMI` pin and the four JTAG/SWD pins (`PB7` and `PC[3:0]`). These five pins are currently the only GPIOs that are protected by the **GPIOCR** register. Because of this, the register type for GPIO Port B7 and GPIO Port C[3:0] is R/W. The default reset value for the **GPIOCR** register is 0x0000.00FF for all GPIO pins, with the exception of the `NMI` pin and the four JTAG/SWD pins (`PB7` and `PC[3:0]`). To ensure that the JTAG port is not accidentally programmed as a GPIO, these four pins default to non-committable. To ensure that the `NMI` pin is not accidentally programmed as the non-maskable interrupt pin, it defaults to non-committable. Because of this, the default reset value of **GPIOCR** for GPIO Port B is 0x0000.007F while the default reset value of GPIOCR for Port C is 0x0000.00F0.

Register 21: GPIO Analog Mode Select (GPIOAMSEL), offset 0x528

Important: This register is only valid for ports D and E.

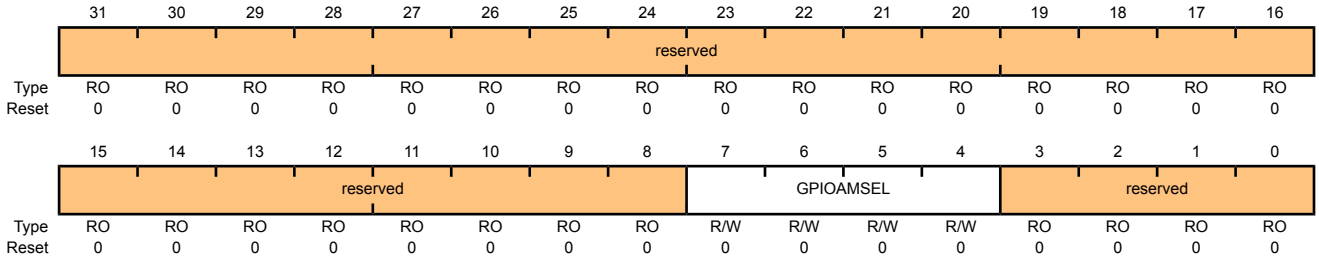
If any pin is to be used as an ADC input, the appropriate bit in **GPIOAMSEL** must be written to 1 to disable the analog isolation circuit.

The **GPIOAMSEL** register controls isolation circuits to the analog side of a unified I/O pad. Because the GPIOs may be driven by a 5V source and affect analog operation, analog circuitry requires isolation from the pins when not used in their analog function.

Each bit of this register controls the isolation circuitry for circuits that share the same pin as the GPIO bit lane.

GPIO Analog Mode Select (GPIOAMSEL)

GPIO Port A (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0x528
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:4	GPIOAMSEL	R/W	0x00	GPIO Analog Mode Select

Value	Description
0	Analog function of the pin is disabled, the isolation is enabled, and the pin is capable of digital functions as specified by the other GPIO configuration registers.
1	Analog function of the pin is enabled, the isolation is disabled, and the pin is capable of analog functions.

Note: This register and bits are required only for GPIO bit lanes that share analog function through a unified I/O pad.

The reset state of this register is 0 for all bit lanes.

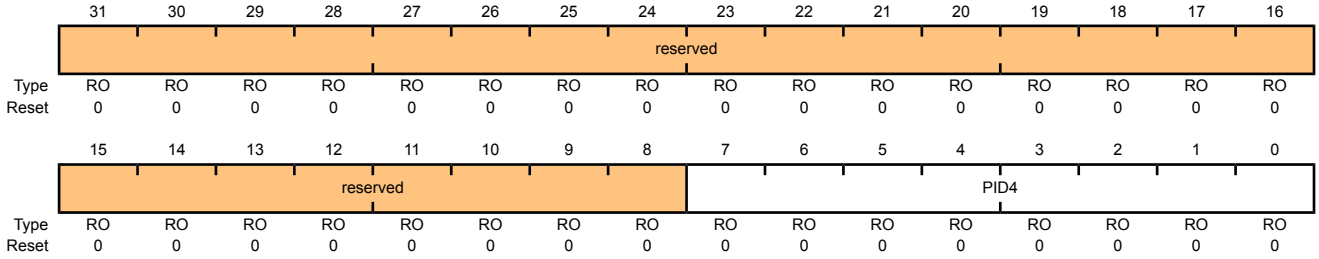
Bit/Field	Name	Type	Reset	Description
3:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 22: 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0xFD0
Type RO, reset 0x0000.0000



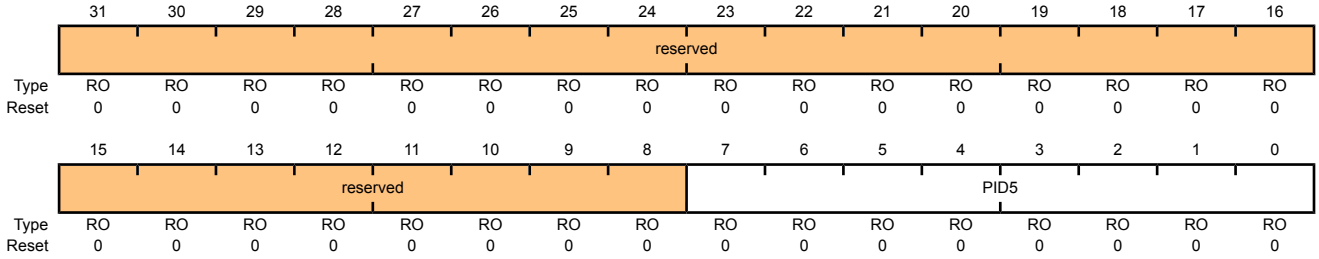
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	GPIO Peripheral ID Register[7:0]

Register 23: 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0xFD4
Type RO, reset 0x0000.0000



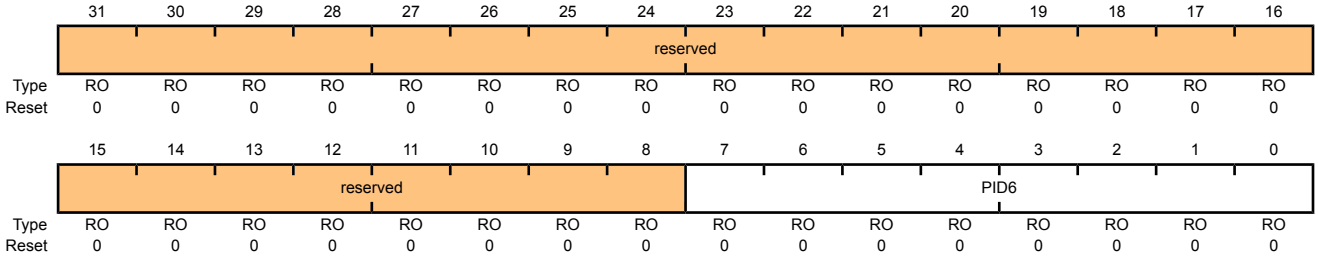
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	GPIO Peripheral ID Register[15:8]

Register 24: 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0xFD8
Type RO, reset 0x0000.0000



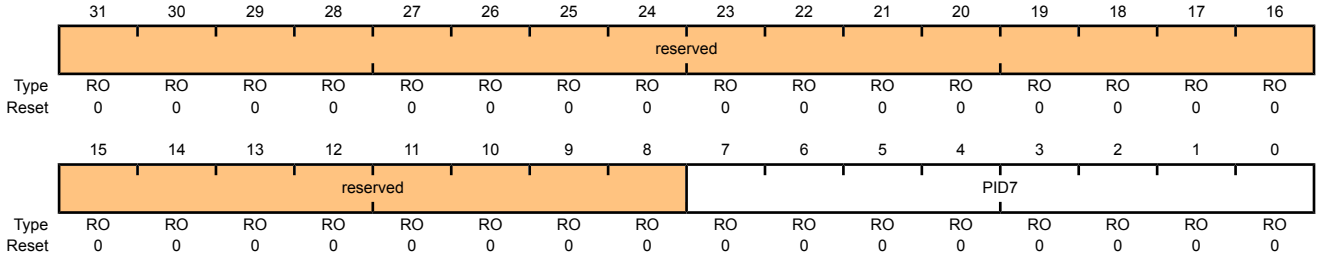
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	GPIO Peripheral ID Register[23:16]

Register 25: 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0xFDC
Type RO, reset 0x0000.0000



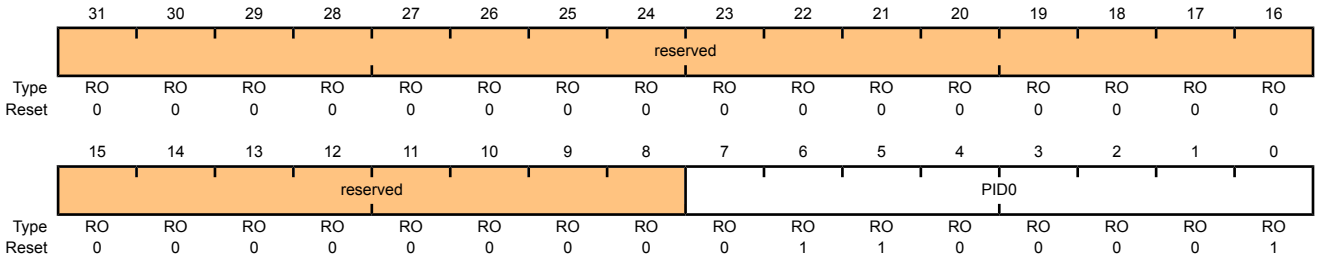
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	GPIO Peripheral ID Register[31:24]

Register 26: 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0xFE0
Type RO, reset 0x0000.0061



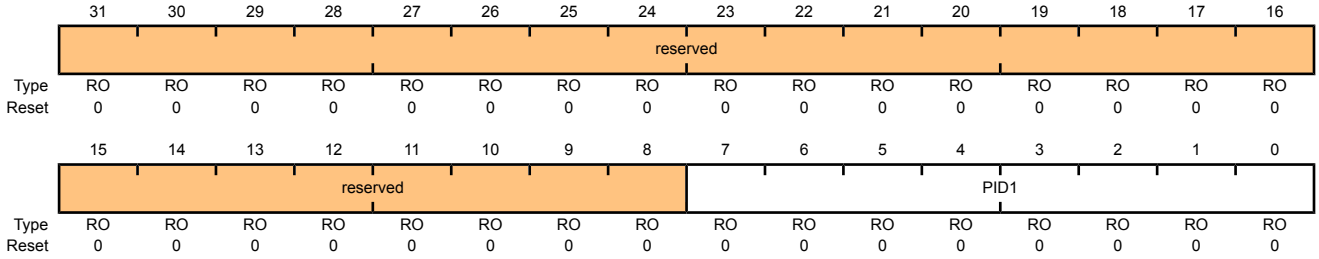
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x61	GPIO Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 27: 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0xFE4
Type RO, reset 0x0000.0000



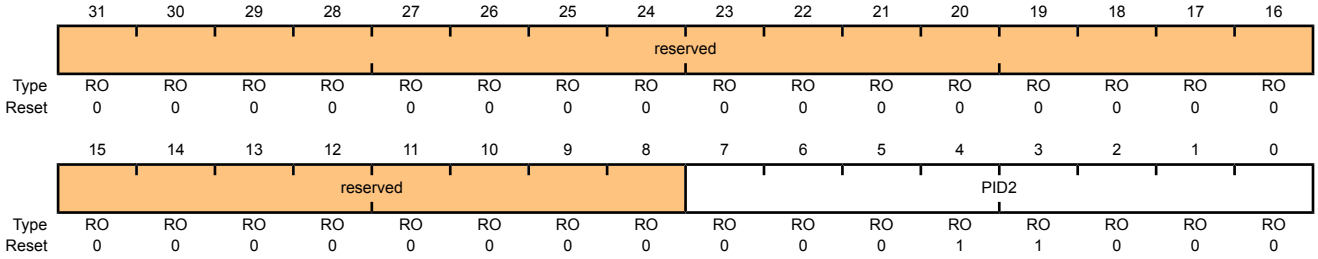
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	GPIO Peripheral ID Register[15:8]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

Register 28: 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0xFE8
Type RO, reset 0x0000.0018



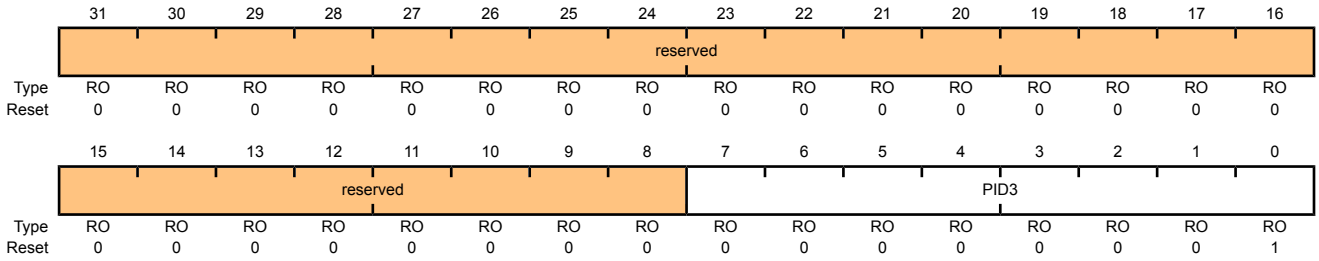
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	GPIO Peripheral ID Register[23:16] Can be used by software to identify the presence of this peripheral.

Register 29: 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0xFEC
Type RO, reset 0x0000.0001



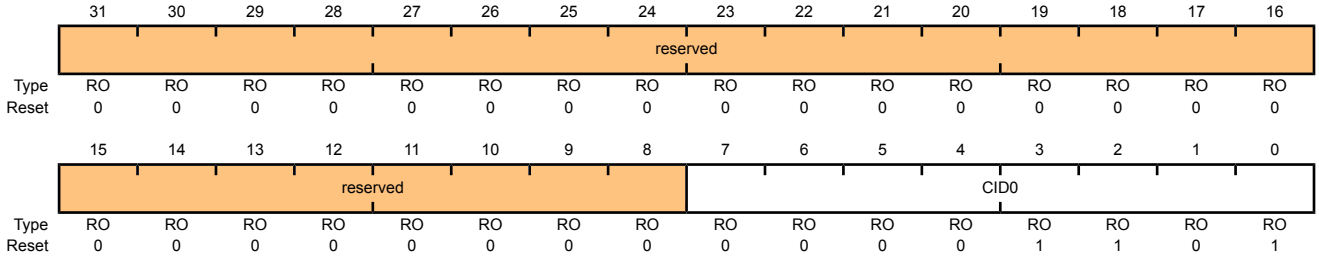
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	GPIO Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

Register 30: 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0xFF0
Type RO, reset 0x0000.000D



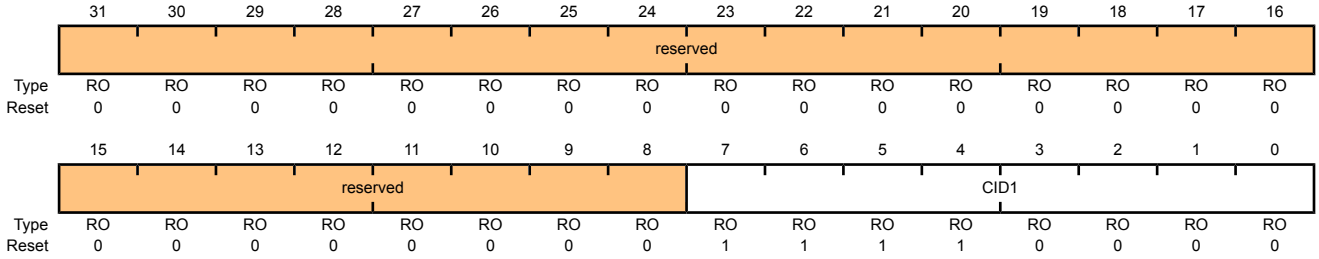
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	GPIO PrimeCell ID Register[7:0] Provides software a standard cross-peripheral identification system.

Register 31: 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0xFF4
Type RO, reset 0x0000.00F0



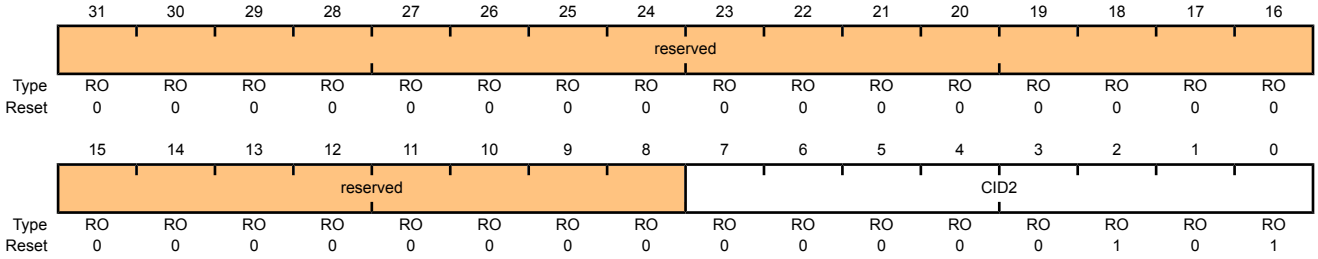
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	GPIO PrimeCell ID Register[15:8] Provides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

Register 32: 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0xFF8
Type RO, reset 0x0000.0005



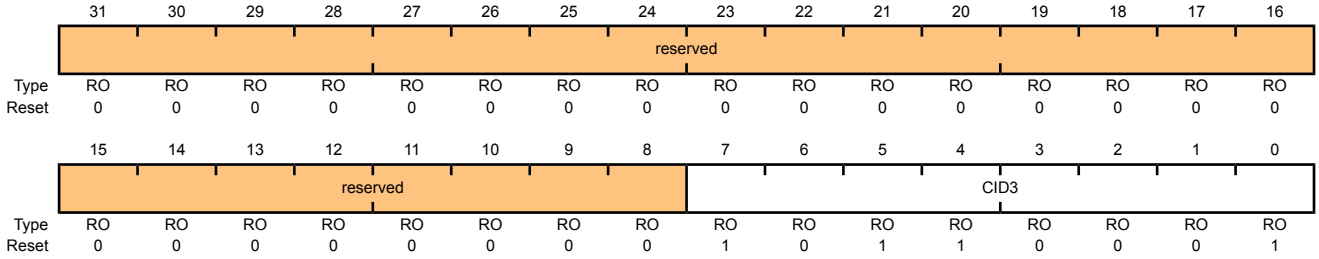
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	GPIO PrimeCell ID Register[23:16] Provides software a standard cross-peripheral identification system.

Register 33: 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 (legacy) base: 0x4000.4000
GPIO Port A (high-speed) base: 0x4005.8000
GPIO Port B (legacy) base: 0x4000.5000
GPIO Port B (high-speed) base: 0x4005.9000
GPIO Port C (legacy) base: 0x4000.6000
GPIO Port C (high-speed) base: 0x4005.A000
GPIO Port D (legacy) base: 0x4000.7000
GPIO Port D (high-speed) base: 0x4005.B000
GPIO Port E (legacy) base: 0x4002.4000
GPIO Port E (high-speed) base: 0x4005.C000
GPIO Port F (legacy) base: 0x4002.5000
GPIO Port F (high-speed) base: 0x4005.D000
GPIO Port G (legacy) base: 0x4002.6000
GPIO Port G (high-speed) base: 0x4005.E000
GPIO Port H (legacy) base: 0x4002.7000
GPIO Port H (high-speed) base: 0x4005.F000
Offset 0xFFC
Type RO, reset 0x0000.00B1



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	GPIO PrimeCell ID Register[31:24] Provides software a standard cross-peripheral identification system.

11 General-Purpose Timers

Programmable timers can be used to count or time external events that drive the Timer input pins. The Stellaris® General-Purpose Timer Module (GPTM) contains four GPTM blocks (Timer0, Timer1, Timer 2, and Timer 3). Each GPTM block provides two 16-bit timers/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).

In addition, timers can be used to trigger analog-to-digital conversions (ADC). The ADC trigger signals from all of the general-purpose timers are ORed together before reaching the ADC module, so only one timer should be used to trigger ADC events.

The GPT Module is one timing resource available on the Stellaris® microcontrollers. Other timer resources include the System Timer (SysTick) (see “System Timer (SysTick)” on page 51) and the PWM timer in the PWM module (see “PWM Timer” on page 662).

The General-Purpose Timers provide the following features:

- Four General-Purpose Timer Modules (GPTM), each of which provides two 16-bit timers. Each GPTM can be configured to operate independently:
  - As a single 32-bit timer
  - As one 32-bit Real-Time Clock (RTC) to event capture
  - For Pulse Width Modulation (PWM)
  - To trigger analog-to-digital conversions
- 32-bit Timer modes
  - Programmable one-shot timer
  - Programmable periodic timer
  - Real-Time Clock when using an external 32.768-KHz clock as the input
  - Software-controlled event stalling (excluding RTC mode)
  - ADC event trigger
- 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
  - User-enabled stalling when the controller asserts CPU Halt flag during debug
  - ADC event trigger
- 16-bit Input Capture modes
  - Input edge count capture

  - Input edge time capture
- 16-bit PWM mode
  - Simple PWM mode with software-programmable output inversion of the PWM signal

11.1 Block Diagram

Note: In Figure 11-1 on page 304, the specific CCP pins available depend on the Stellaris® device. See Table 11-1 on page 304 for the available CCPs.

Figure 11-1. GPTM Module Block Diagram

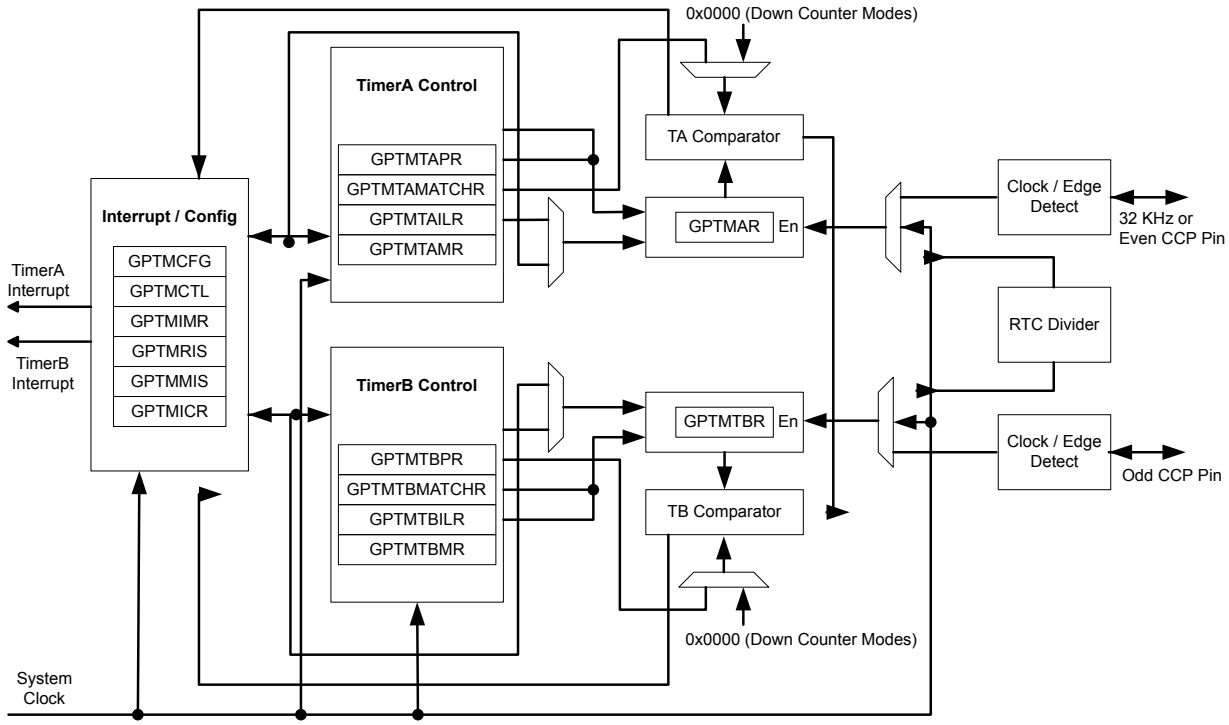


Table 11-1. Available CCP Pins

Timer	16-Bit Up/Down Counter	Even CCP Pin	Odd CCP Pin
Timer 0	TimerA	CCP0	-
	TimerB	-	CCP1
Timer 1	TimerA	CCP2	-
	TimerB	-	CCP3
Timer 2	TimerA	CCP4	-
	TimerB	-	-
Timer 3	TimerA	-	-
	TimerB	-	-

11.2 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, 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 315), the **GPTM TimerA Mode (GPTMTAMR)** register (see page 316), and the **GPTM TimerB Mode (GPTMTBMR)** register (see page 318). When in one of the 32-bit modes, the timer can only act as a 32-bit timer. However, when configured in 16-bit mode, the GPTM can have its two 16-bit timers configured in any combination of the 16-bit modes.

11.2.1 GPTM Reset Conditions

After reset has been applied to the GPTM module, the module is in an inactive state, and all control registers are cleared and in their default states. Counters TimerA and TimerB are initialized to 0xFFFF, along with their corresponding load registers: the **GPTM TimerA Interval Load (GPTMTAILR)** register (see page 329) and the **GPTM TimerB Interval Load (GPTMTBILR)** register (see page 330). The prescale counters are initialized to 0x00: the **GPTM TimerA Prescale (GPTMTAPR)** register (see page 333) and the **GPTM TimerB Prescale (GPTMTBPR)** register (see page 334).

11.2.2 32-Bit Timer Operating Modes

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

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

- **GPTM TimerA Interval Load (GPTMTAILR)** register [15:0], see page 329
- **GPTM TimerB Interval Load (GPTMTBILR)** register [15:0], see page 330
- **GPTM TimerA (GPTMTAR)** register [15:0], see page 335
- **GPTM TimerB (GPTMTBR)** register [15:0], see page 336

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]
```

11.2.2.1 32-Bit One-Shot/Periodic Timer Mode

In 32-bit one-shot and periodic timer modes, the concatenated versions of the 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 316), 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 320), 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 triggers when it reaches the 0x000.0000 state. The GPTM sets the TATORIS bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register (see page 325), and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register (see page 327). If the time-out interrupt is enabled in the **GPTM Interrupt Mask (GPTIMR)** register (see page 323), the GPTM also sets the TATOMIS bit in the **GPTM Masked Interrupt Status (GPTMMIS)** register (see page 326). The ADC trigger 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.

11.2.2.2 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 331) 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**.

11.2.3 16-Bit Timer Operating Modes

The GPTM is placed into global 16-bit mode by writing a value of 0x4 to the **GPTM Configuration (GPTMCFG)** register (see page 315). 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.

11.2.3.1 16-Bit One-Shot/Periodic Timer Mode

In 16-bit one-shot and periodic timer modes, the timer is configured as a 16-bit 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 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 ADC trigger is enabled by setting the TnOTE bit in the **GPTMCTL** register.

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 11-2. 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.

11.2.3.2 16-Bit Input Edge Count Mode

Note: For rising-edge detection, the input signal must be High for at least two system clock periods following the rising edge. Similarly, for falling-edge detection, the input signal must be Low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the system frequency.

Note: The prescaler is not available in 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.

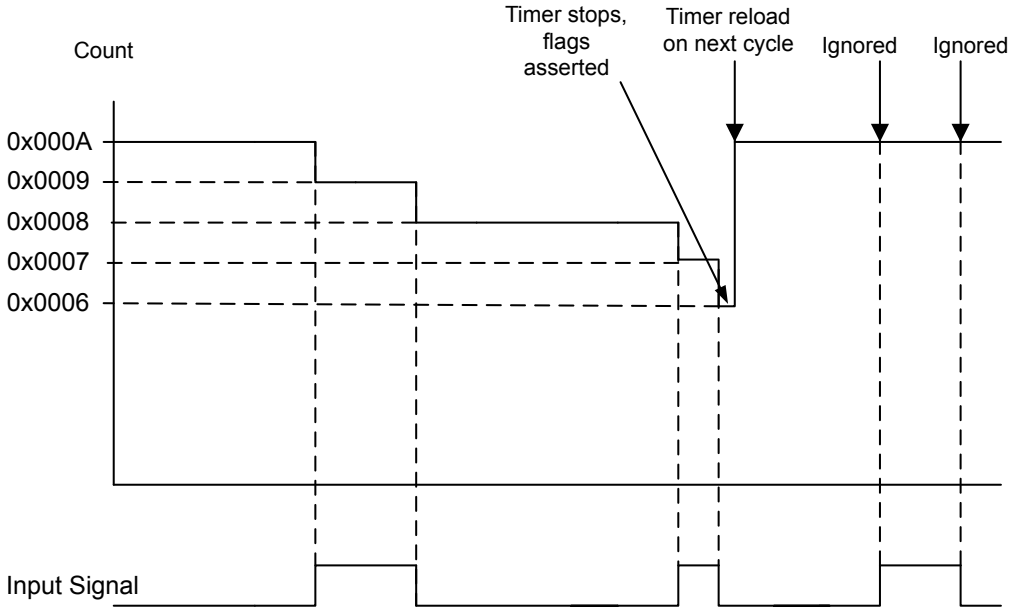
When software writes the TnEN bit in the **GPTM Control (GPTMCTL)** register, the timer is enabled for event capture. Each input event on the CCP pin decrements the counter by 1 until the event count matches **GPTMTnMATCHR**. When the counts match, the GPTM asserts the CnMRIS bit in the **GPTMRIS** register (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 11-2 on page 308 shows how input edge count mode works. In this case, the timer start value is set to **GPTMnILR** =0x000A and the match value is set to **GPTMnMATCHR** =0x0006 so that four edge events are counted. The counter is configured to detect both edges of the input signal.

Note that the last two edges are not counted since the timer automatically clears the TnEN bit after the current count matches the value in the **GPTMnMR** register.

Figure 11-2. 16-Bit Input Edge Count Mode Example



11.2.3.3 16-Bit Input Edge Time Mode

Note: For rising-edge detection, the input signal must be High for at least two system clock periods following the rising edge. Similarly, for falling edge detection, the input signal must be Low for at least two system clock periods following the falling edge. Based on this criteria, the maximum input frequency for edge detection is 1/4 of the system frequency.

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 either rising or falling edges, but not both. The timer is placed into Edge Time mode by setting the TnCMR bit in the **GPTMTnMR** register, and the type of event that the timer captures is determined by the TnEVENT fields of the **GPTMCnTL** register.

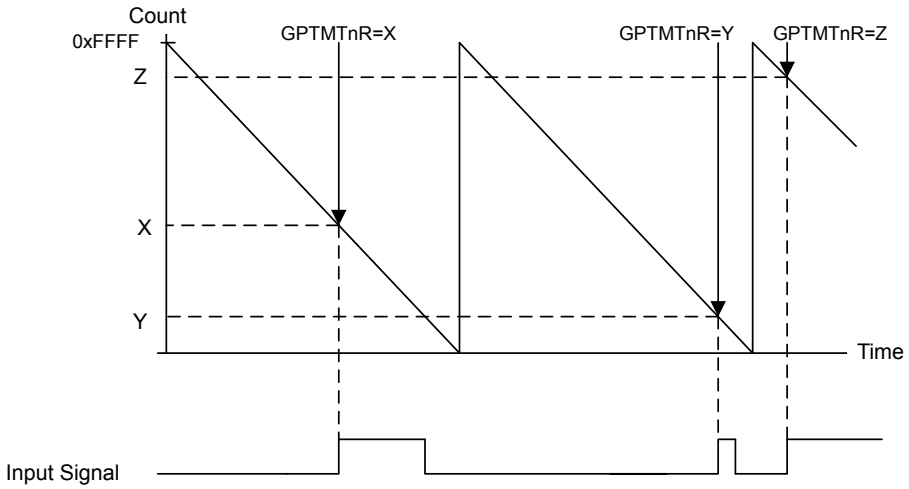
When software writes the TnEN bit in the **GPTMCTL** register, the timer is enabled for event capture. When the selected input event is detected, the current Tn counter value is captured in the **GPTMTnR** register and is available to be read by the controller. The GPTM then asserts the CnERIS bit (and 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 11-3 on page 309 shows how input edge timing mode works. In the diagram, it is assumed that the start value of the timer is the default value of 0xFFFF, and the timer is configured to capture rising edge events.

Each time a rising edge event is detected, the current count value is loaded into the **GPTMTnR** register, and is held there until another rising edge is detected (at which point the new count value is loaded into **GPTMTnR**).

Figure 11-3. 16-Bit Input Edge Time Mode Example



11.2.3.4 16-Bit PWM Mode

Note: The prescaler is not available in 16-Bit PWM mode.

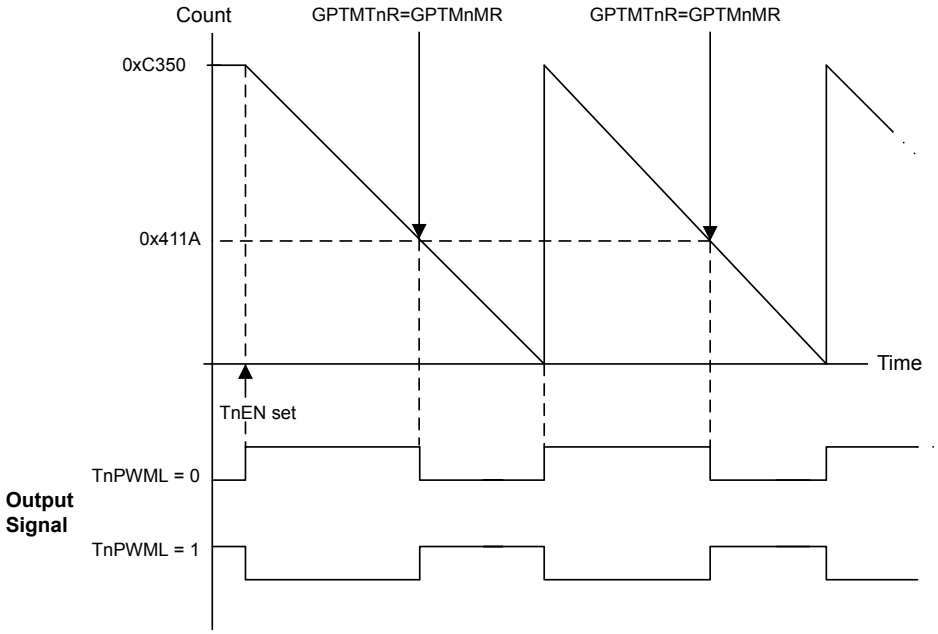
The GPTM supports a simple PWM generation mode. In PWM mode, the timer is configured as a down-counter with a start value (and thus period) defined by **GPTMTnILR**. PWM mode is enabled with the **GPTMTnMR** register by setting the TnAMS bit to 0x1, the TnCMR bit to 0x0, and the TnMR field to 0x2.

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

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

Figure 11-4 on page 310 shows how to generate an output PWM with a 1-ms period and a 66% duty cycle assuming a 50-MHz input clock and **TnPWML** =0 (duty cycle would be 33% for the **TnPWML** =1 configuration). For this example, the start value is **GPTMnIRL**=0xC350 and the match value is **GPTMnMR**=0x411A.

Figure 11-4. 16-Bit PWM Mode Example



11.3 Initialization and Configuration

To use the general-purpose timers, the peripheral clock must be enabled by setting the TIMER0, TIMER1, TIMER2, and TIMER3 bits in the **RCGC1** register.

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

11.3.1 32-Bit One-Shot/Periodic Timer Mode

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

1. Ensure the timer is disabled (the TAEN bit in the **GPTMCTL** register is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x0.
3. 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 311. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode does not stop counting after it times out.

11.3.2 32-Bit Real-Time Clock (RTC) Mode

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

1. Ensure the timer is disabled (the TAEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x1.
3. Write the desired match value to the **GPTM 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.

11.3.3 16-Bit One-Shot/Periodic Timer Mode

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

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration Register (GPTMCFG)** with a value of 0x4.
3. Set the TnMR field in the **GPTM Timer Mode (GPTMTnMR)** register:
   a. Write a value of 0x1 for One-Shot mode.
   b. Write a value of 0x2 for Periodic mode.
4. 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 311. To re-enable the timer, repeat the sequence. A timer configured in Periodic mode does not stop counting after it times out.

11.3.4 16-Bit Input Edge Count Mode

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

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x4.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the 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 312 through step 9 on page 312.

11.3.5 16-Bit Input Edge Timing Mode

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

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register with a value of 0x4.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, write the TnCMR field to 0x1 and the TnMR field to 0x3.
4. Configure the type of event that the timer captures by writing the TnEVENT field of the **GPTM Control (GPTMCTL)** register.
5. 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.

11.3.6 16-Bit PWM Mode

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

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

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

11.4 Register Map

Table 11-3 on page 313 lists the GPTM registers. The offset listed is a hexadecimal increment to the register's address, relative to that timer's base address:

- Timer0: 0x4003.0000
- Timer1: 0x4003.1000
- Timer2: 0x4003.2000
- Timer3: 0x4003.3000

Table 11-3. Timers Register Map

Offset	Name	Type	Reset	Description	See page
0x000	GPTMCFG	R/W	0x0000.0000	GPTM Configuration	315
0x004	GPTMTAMR	R/W	0x0000.0000	GPTM TimerA Mode	316
0x008	GPTMTBMR	R/W	0x0000.0000	GPTM TimerB Mode	318
0x00C	GPTMCTL	R/W	0x0000.0000	GPTM Control	320
0x018	GPTMIMR	R/W	0x0000.0000	GPTM Interrupt Mask	323
0x01C	GPTMRIS	RO	0x0000.0000	GPTM Raw Interrupt Status	325

Offset	Name	Type	Reset	Description	See page
0x020	GPTMMIS	RO	0x0000.0000	GPTM Masked Interrupt Status	326
0x024	GPTMICR	W1C	0x0000.0000	GPTM Interrupt Clear	327
0x028	GPTMTAILR	R/W	0x0000.FFFF (16-bit mode) 0xFFFF.FFFF (32-bit mode)	GPTM TimerA Interval Load	329
0x02C	GPTMTBILR	R/W	0x0000.FFFF	GPTM TimerB Interval Load	330
0x030	GPTMTAMATCHR	R/W	0x0000.FFFF (16-bit mode) 0xFFFF.FFFF (32-bit mode)	GPTM TimerA Match	331
0x034	GPTMTBMATCHR	R/W	0x0000.FFFF	GPTM TimerB Match	332
0x038	GPTMTAPR	R/W	0x0000.0000	GPTM TimerA Prescale	333
0x03C	GPTMTBPR	R/W	0x0000.0000	GPTM TimerB Prescale	334
0x048	GPTMTAR	RO	0x0000.FFFF (16-bit mode) 0xFFFF.FFFF (32-bit mode)	GPTM TimerA	335
0x04C	GPTMTBR	RO	0x0000.FFFF	GPTM TimerB	336

11.5 Register Descriptions

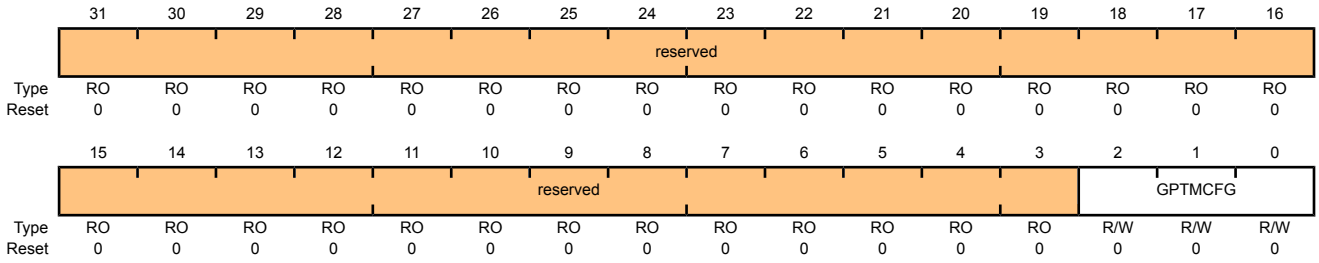
The remainder of this section lists and describes the GPTM registers, in numerical order by address offset.

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

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

GPTM Configuration (GPTMCFG)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x000
Type R/W, reset 0x0000.0000



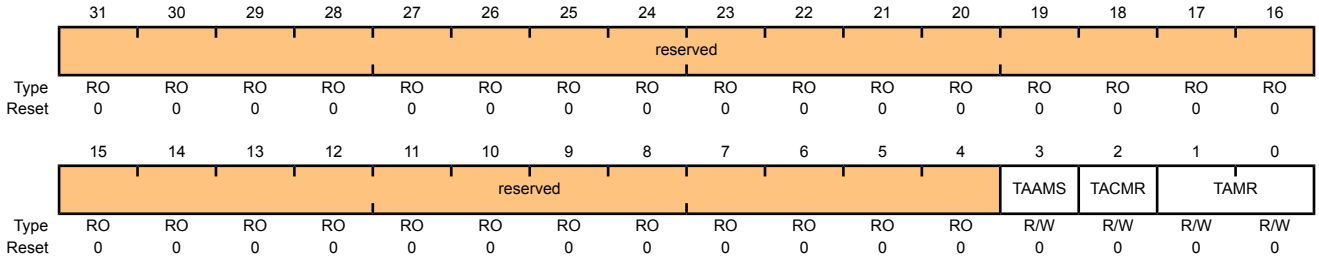
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	GPTMCFG	R/W	0x0	GPTM Configuration The GPTMCFG values are defined as follows: Value Description 0x0 32-bit timer configuration. 0x1 32-bit real-time clock (RTC) counter configuration. 0x2 Reserved 0x3 Reserved 0x4-0x7 16-bit timer configuration, function is controlled by bits 1:0 of **GPTMTAMR** and **GPTMTBMR**.

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

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

GPTM TimerA Mode (GPTMTAMR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x004
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	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

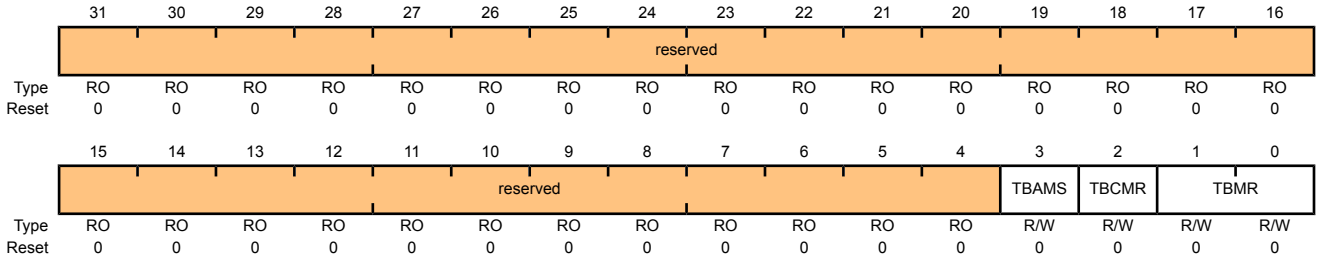
Bit/Field	Name	Type	Reset	Description
1:0	TAMR	R/W	0x0	GPTM TimerA Mode The TAMR values are defined as follows: Value Description 0x0 Reserved 0x1 One-Shot Timer mode 0x2 Periodic Timer mode 0x3 Capture mode The Timer mode is based on the timer configuration defined by bits 2:0 in the **GPTMCFG** register (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
Timer3 base: 0x4003.3000
Offset 0x008
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	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

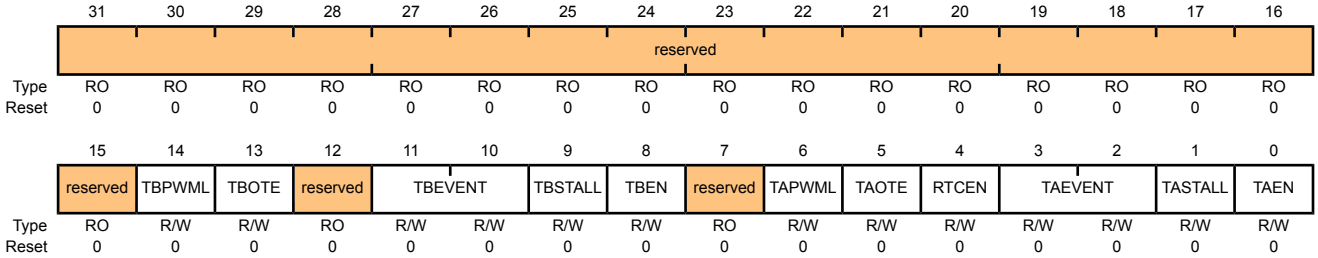
Bit/Field	Name	Type	Reset	Description
1:0	TBMR	R/W	0x0	GPTM TimerB Mode The TBMR values are defined as follows: Value Description 0x0 Reserved 0x1 One-Shot Timer mode 0x2 Periodic Timer mode 0x3 Capture mode The timer mode is based on the timer configuration defined by bits 2:0 in the **GPTMCFG** register. 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. The output trigger can be used to initiate transfers on the ADC module.

GPTM Control (GPTMCTL)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x00C
Type R/W, reset 0x0000.0000



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 ADC trigger is disabled. 1 The output TimerB ADC trigger is enabled. In addition, the ADC must be enabled and the timer selected as a trigger source with the EMn bit in the **ADCEMUX** register (see page 376).
12	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
11:10	TBEVENT	R/W	0x0	GPTM TimerB Event Mode The TBEVENT values are defined as follows: Value Description 0x0 Positive edge 0x1 Negative edge 0x2 Reserved 0x3 Both edges
9	TBSTALL	R/W	0	GPTM 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 ADC trigger is disabled. 1 The output TimerA ADC trigger is enabled. In addition, the ADC must be enabled and the timer selected as a trigger source with the EMn bit in the **ADCEMUX** register (see page 376).

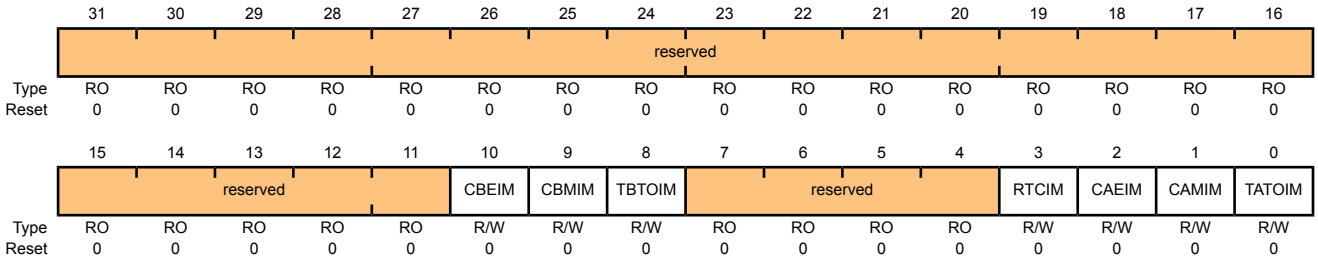
Bit/Field	Name	Type	Reset	Description
4	RTCEN	R/W	0	GPTM RTC Enable The RTCEN values are defined as follows: Value Description 0 RTC counting is disabled. 1 RTC counting is enabled.
3:2	TAEVENT	R/W	0x0	GPTM TimerA Event Mode The TAEVENT values are defined as follows: Value Description 0x0 Positive edge 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
Timer3 base: 0x4003.3000
Offset 0x018
Type R/W, reset 0x0000.0000



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.

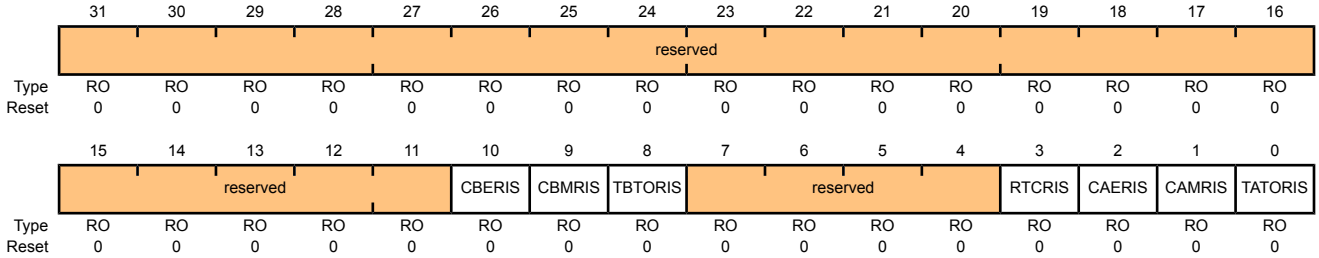
Bit/Field	Name	Type	Reset	Description
3	RTCIM	R/W	0	GPTM RTC Interrupt Mask The RTCIM values are defined as follows: Value Description 0 Interrupt is disabled. 1 Interrupt is enabled.
2	CAEIM	R/W	0	GPTM CaptureA Event Interrupt Mask The CAEIM values are defined as follows: Value Description 0 Interrupt 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
Timer3 base: 0x4003.3000
Offset 0x01C
Type RO, reset 0x0000.0000



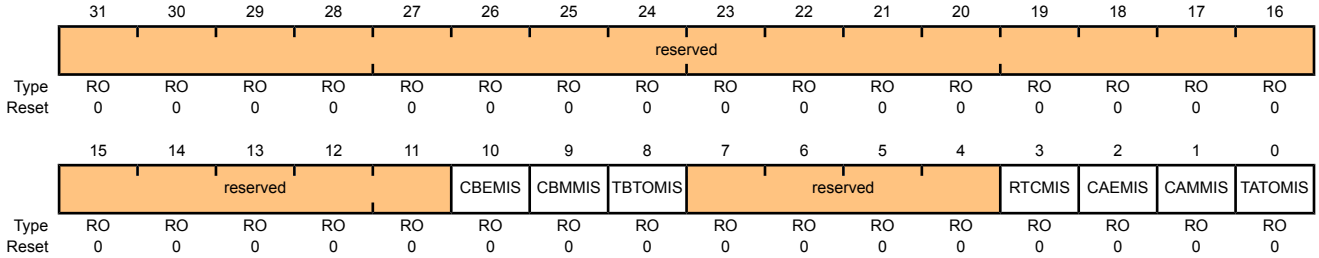
Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBERIS	RO	0	GPTM CaptureB Event Raw Interrupt This 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
Timer3 base: 0x4003.3000
Offset 0x020
Type RO, reset 0x0000.0000



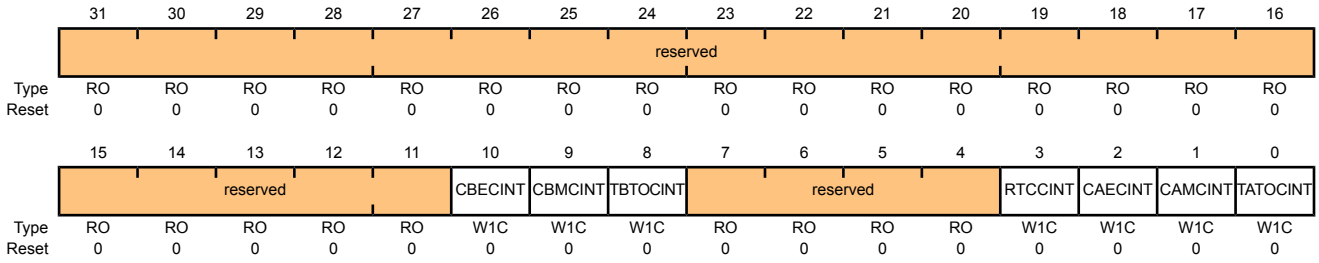
Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	CBEMIS	RO	0	GPTM CaptureB Event Masked Interrupt 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
Timer3 base: 0x4003.3000
Offset 0x024
Type W1C, reset 0x0000.0000



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.

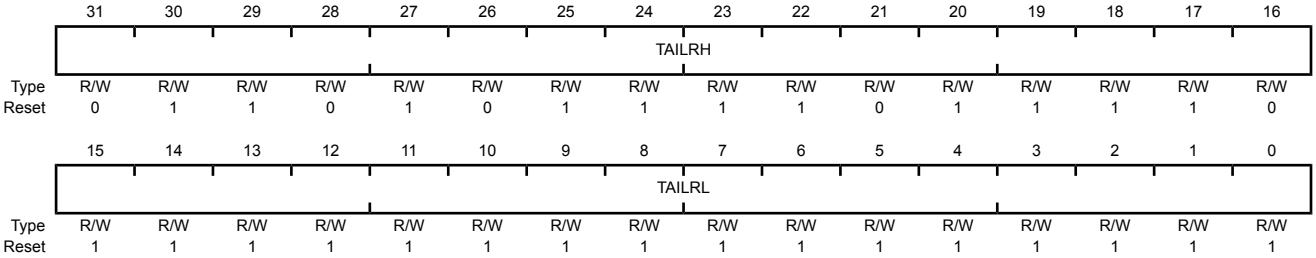
Bit/Field	Name	Type	Reset	Description
3	RTCCINT	W1C	0	GPTM RTC Interrupt Clear The RTCCINT values are defined as follows: Value Description 0 The interrupt is unaffected. 1 The interrupt is cleared.
2	CAECINT	W1C	0	GPTM CaptureA Event Interrupt Clear The CAECINT values are defined as follows: Value Description 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
Timer3 base: 0x4003.3000
Offset 0x028
Type R/W, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)



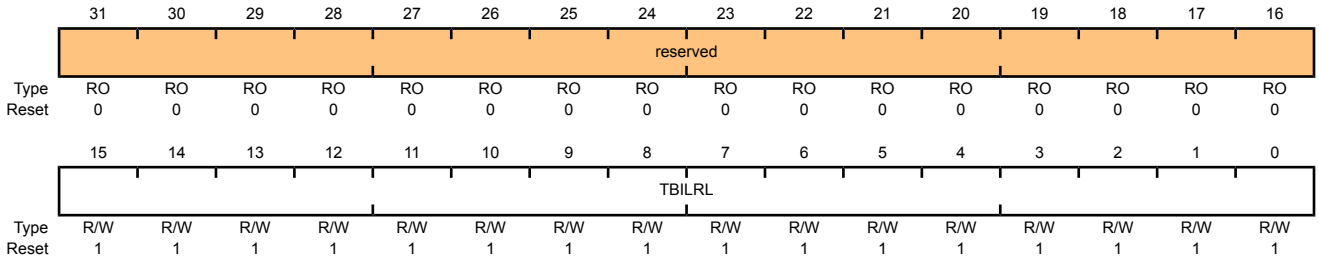
Bit/Field	Name	Type	Reset	Description
31:16	TAILRH	R/W	0xFFFF (32-bit mode) 0x0000 (16-bit mode)	GPTM TimerA Interval Load Register High When configured for 32-bit mode via the **GPTMCFG** register, the **GPTM TimerB Interval Load (GPTMTBILR)** register loads this value on a write. A read returns the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

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

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

GPTM TimerB Interval Load (GPTMTBILR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x02C
Type R/W, reset 0x0000.FFFF



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	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
Timer3 base: 0x4003.3000
Offset 0x030
Type R/W, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)

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

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

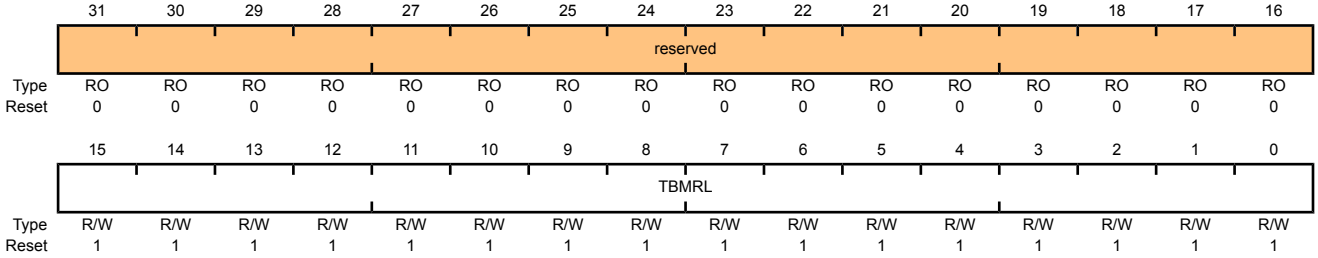
Bit/Field	Name	Type	Reset	Description
31:16	TAMRH	R/W	0xFFFF (32-bit mode) 0x0000 (16-bit mode)	GPTM TimerA Match Register High When configured for 32-bit Real-Time Clock (RTC) mode via the **GPTMCFG** register, this value is compared to the upper half of **GPTMTAR**, to determine match events. In 16-bit 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 16-bit PWM and Input Edge Count modes.

GPTM TimerB Match (GPTMTBMATCHR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x034
Type R/W, reset 0x0000.FFFF



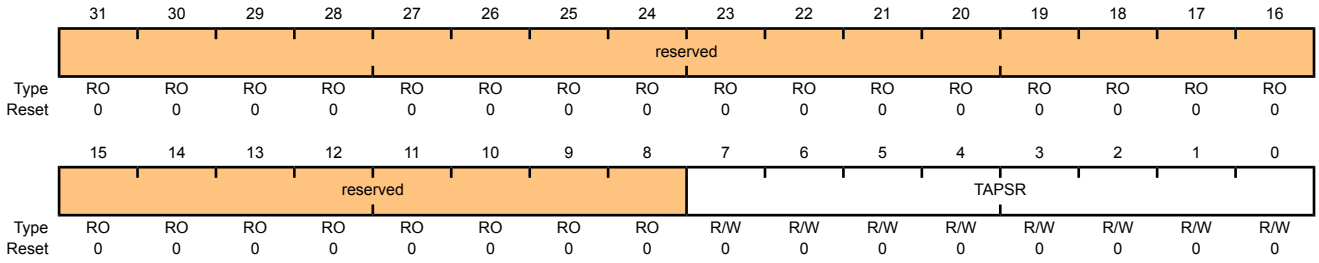
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TBMRL	R/W	0xFFFF	GPTM TimerB Match Register 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
Timer3 base: 0x4003.3000
Offset 0x038
Type R/W, reset 0x0000.0000



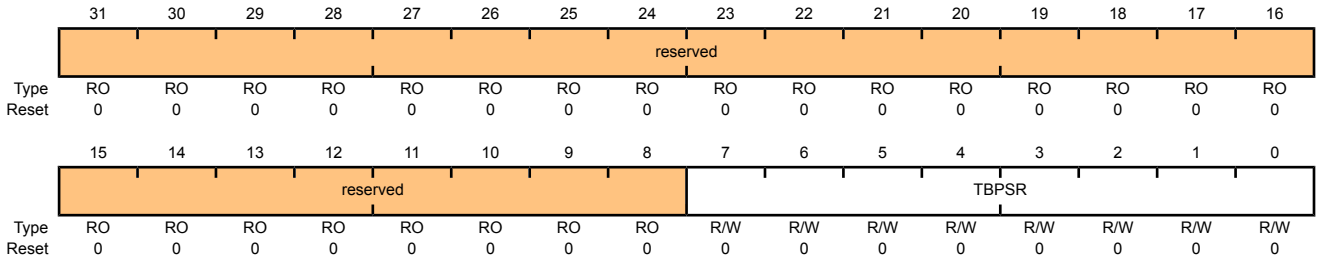
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TAPSR	R/W	0x00	GPTM TimerA Prescale The register loads this value on a write. A read returns the current value of the register. Refer to Table 11-2 on page 307 for more details and an example.

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

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

GPTM TimerB Prescale (GPTMTBPR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x03C
Type R/W, reset 0x0000.0000



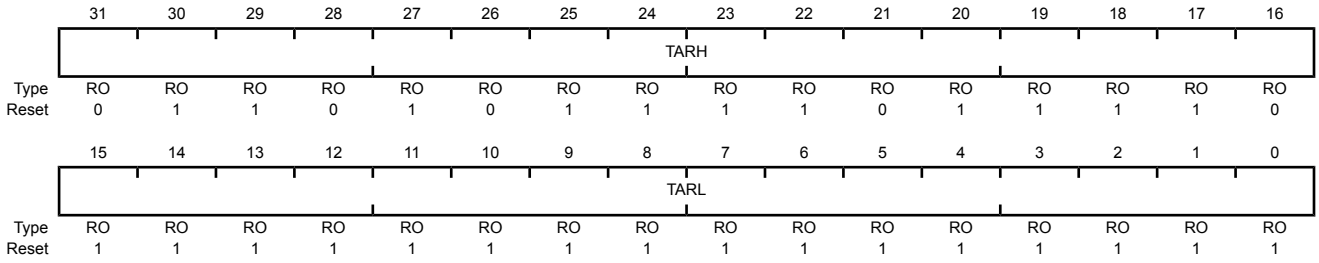
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TBPSR	R/W	0x00	GPTM TimerB Prescale The register loads this value on a write. A read returns the current value of this register. Refer to Table 11-2 on page 307 for more details and an example.

Register 15: GPTM TimerA (GPTMTAR), offset 0x048

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

GPTM TimerA (GPTMTAR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x048
Type RO, reset 0x0000.FFFF (16-bit mode) and 0xFFFF.FFFF (32-bit mode)



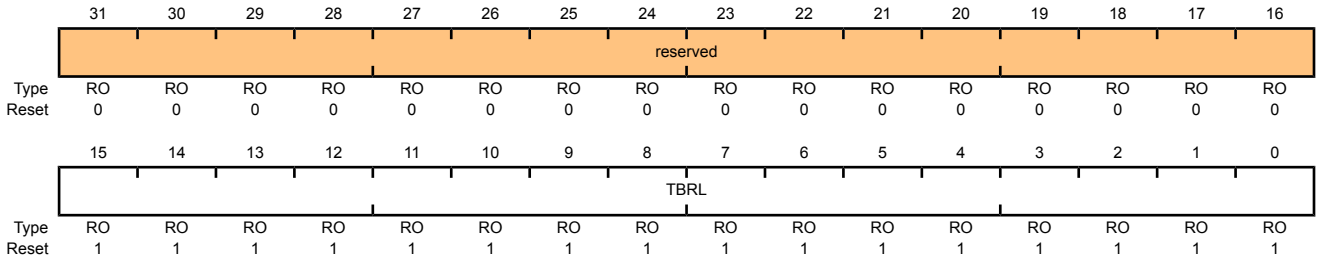
Bit/Field	Name	Type	Reset	Description
31:16	TARH	RO	0xFFFF (32-bit mode) 0x0000 (16-bit mode)	GPTM TimerA Register High If the **GPTMCFG** is in a 32-bit mode, TimerB value is read. If the **GPTMCFG** is in a 16-bit mode, this is read as zero.
15:0	TARL	RO	0xFFFF	GPTM TimerA Register Low A read returns 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 16: GPTM TimerB (GPTMTBR), offset 0x04C

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

GPTM TimerB (GPTMTBR)

Timer0 base: 0x4003.0000
Timer1 base: 0x4003.1000
Timer2 base: 0x4003.2000
Timer3 base: 0x4003.3000
Offset 0x04C
Type RO, reset 0x0000.FFFF



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	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.

12 Watchdog Timer

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

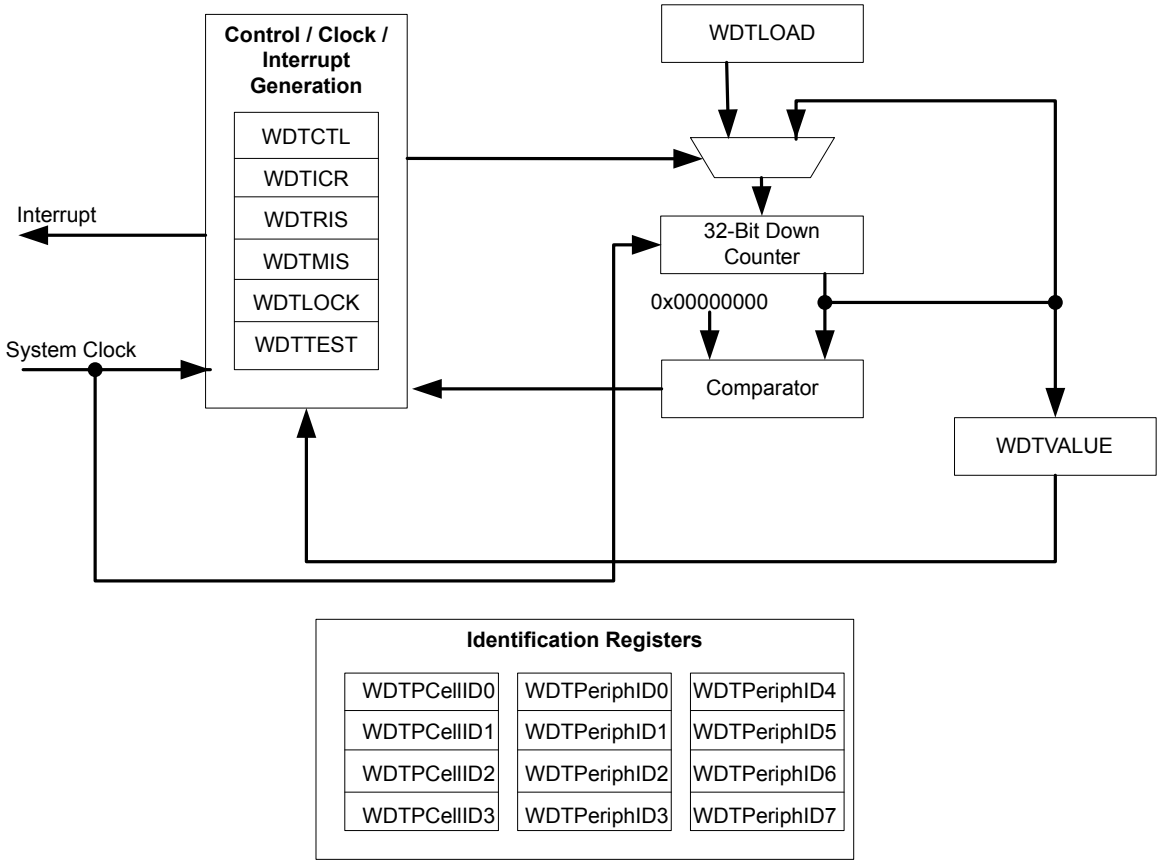
The Stellaris® Watchdog Timer module has the following features:

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

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

12.1 Block Diagram

Figure 12-1. WDT Module Block Diagram



12.2 Functional Description

The Watchdog Timer module generates the first time-out signal when the 32-bit counter reaches the zero state after being enabled; enabling the counter also enables the watchdog timer interrupt. After the first time-out event, the 32-bit counter is re-loaded with the value of the **Watchdog Timer Load (WDTLOAD)** register, and the timer resumes counting down from that value. Once the Watchdog Timer has been configured, the **Watchdog Timer Lock (WDTLOCK)** register is written, which prevents the timer configuration from being inadvertently altered by software.

If the timer counts down to its zero state again before the first time-out interrupt is cleared, and the reset signal has been enabled (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.

12.3 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.

12.4 Register Map

Table 12-1 on page 339 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 12-1. Watchdog Timer Register Map

Offset	Name	Type	Reset	Description	See page
0x000	WDTLOAD	R/W	0xFFFF.FFFF	Watchdog Load	341
0x004	WDTVALUE	RO	0xFFFF.FFFF	Watchdog Value	342
0x008	WDTCTL	R/W	0x0000.0000	Watchdog Control	343
0x00C	WDTICR	WO	-	Watchdog Interrupt Clear	344
0x010	WDTRIS	RO	0x0000.0000	Watchdog Raw Interrupt Status	345
0x014	WDTMIS	RO	0x0000.0000	Watchdog Masked Interrupt Status	346
0x418	WDTTEST	R/W	0x0000.0000	Watchdog Test	347
0xC00	WDTLOCK	R/W	0x0000.0000	Watchdog Lock	348
0xFD0	WDTPeriphID4	RO	0x0000.0000	Watchdog Peripheral Identification 4	349
0xFD4	WDTPeriphID5	RO	0x0000.0000	Watchdog Peripheral Identification 5	350
0xFD8	WDTPeriphID6	RO	0x0000.0000	Watchdog Peripheral Identification 6	351
0xFDC	WDTPeriphID7	RO	0x0000.0000	Watchdog Peripheral Identification 7	352
0xFE0	WDTPeriphID0	RO	0x0000.0005	Watchdog Peripheral Identification 0	353
0xFE4	WDTPeriphID1	RO	0x0000.0018	Watchdog Peripheral Identification 1	354
0xFE8	WDTPeriphID2	RO	0x0000.0018	Watchdog Peripheral Identification 2	355

Offset	Name	Type	Reset	Description	See page
0xFEC	WDTPeriphID3	RO	0x0000.0001	Watchdog Peripheral Identification 3	356
0xFF0	WDTPCellID0	RO	0x0000.000D	Watchdog PrimeCell Identification 0	357
0xFF4	WDTPCellID1	RO	0x0000.00F0	Watchdog PrimeCell Identification 1	358
0xFF8	WDTPCellID2	RO	0x0000.0005	Watchdog PrimeCell Identification 2	359
0xFFC	WDTPCellID3	RO	0x0000.00B1	Watchdog PrimeCell Identification 3	360

12.5 Register Descriptions

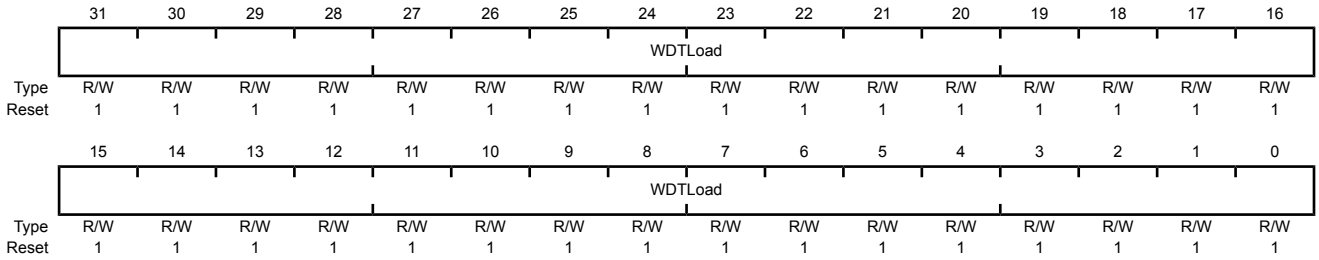
The remainder of this section lists and describes the WDT registers, in numerical order by address offset.

Register 1: Watchdog Load (WDTLOAD), offset 0x000

This register is the 32-bit interval value used by the 32-bit counter. When this register is written, the value is immediately loaded and the counter restarts counting down from the new value. If the **WDTLOAD** register is loaded with 0x0000.0000, an interrupt is immediately generated.

Watchdog Load (WDTLOAD)

Base 0x4000.0000
Offset 0x000
Type R/W, reset 0xFFFF.FFFF



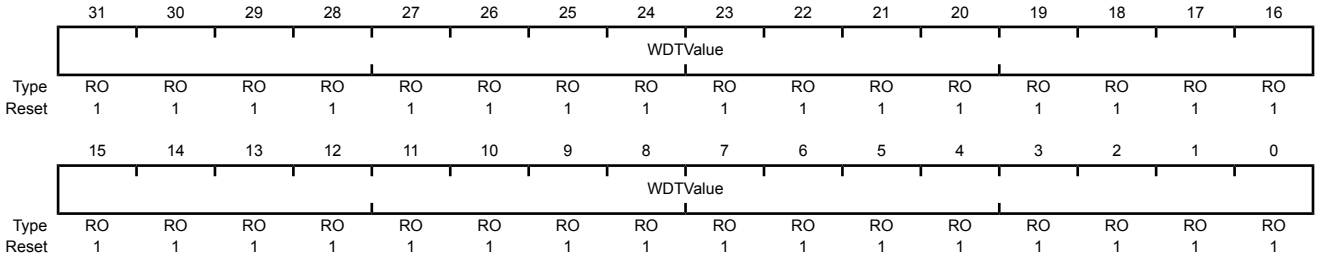
Bit/Field	Name	Type	Reset	Description
31:0	WDTLoad	R/W	0xFFFF.FFFF	Watchdog Load Value

Register 2: Watchdog Value (WDTVALUE), offset 0x004

This register contains the current count value of the timer.

Watchdog Value (WDTVALUE)

Base 0x4000.0000
Offset 0x004
Type RO, reset 0xFFFF.FFFF



Bit/Field	Name	Type	Reset	Description
31:0	WDTValue	RO	0xFFFF.FFFF	Watchdog Value Current value of the 32-bit down counter.

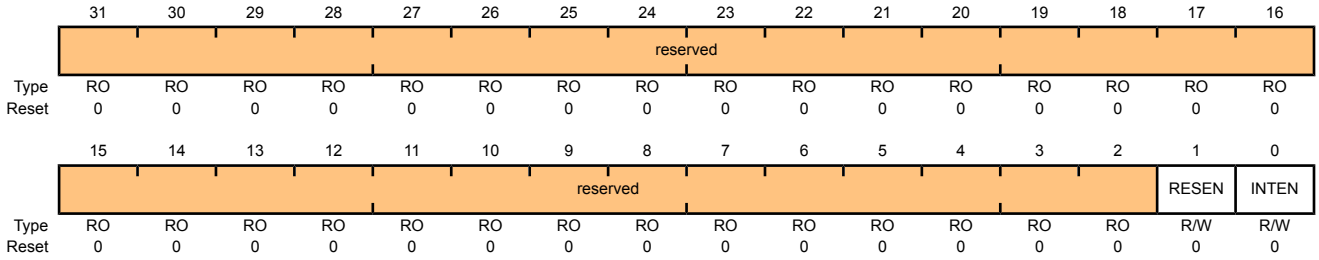
Register 3: Watchdog Control (WDTCTL), offset 0x008

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



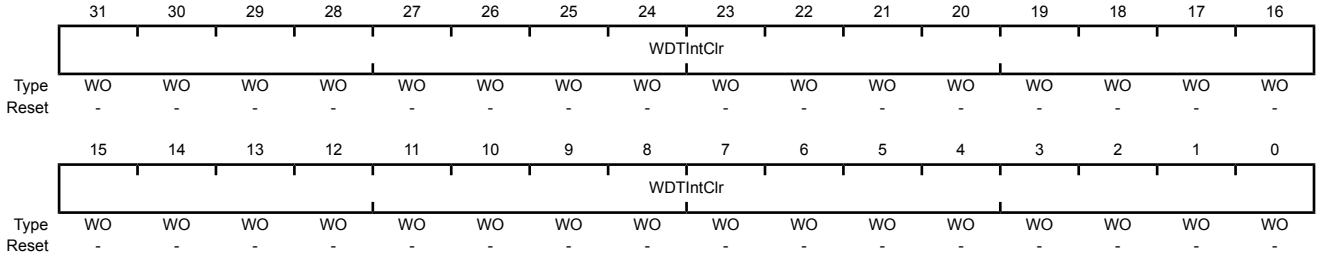
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	RESEN	R/W	0	Watchdog Reset Enable The RESEN values 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 -



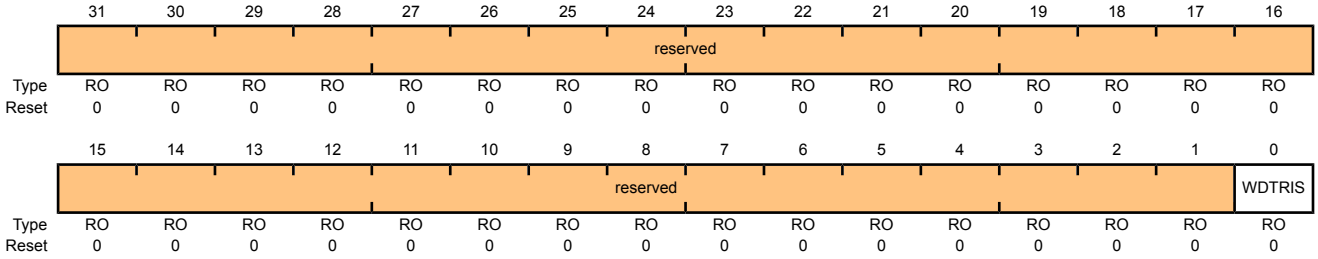
Bit/Field	Name	Type	Reset	Description
31:0	WDTIntClr	WO	-	Watchdog Interrupt Clear

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



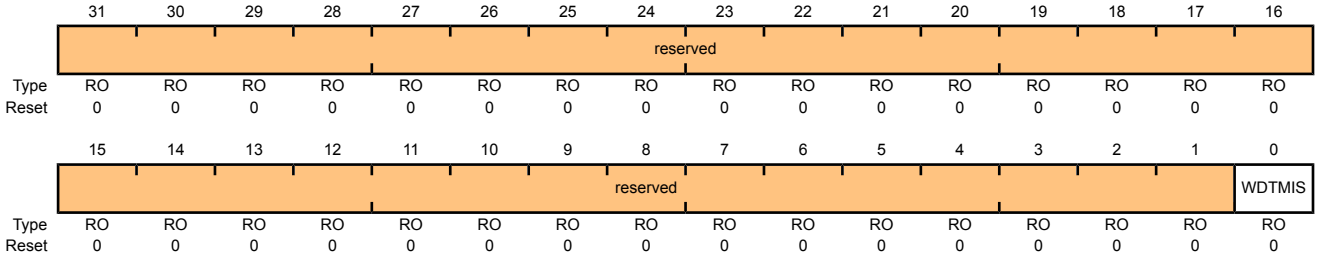
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WDTRIS	RO	0	Watchdog Raw Interrupt Status Gives the 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



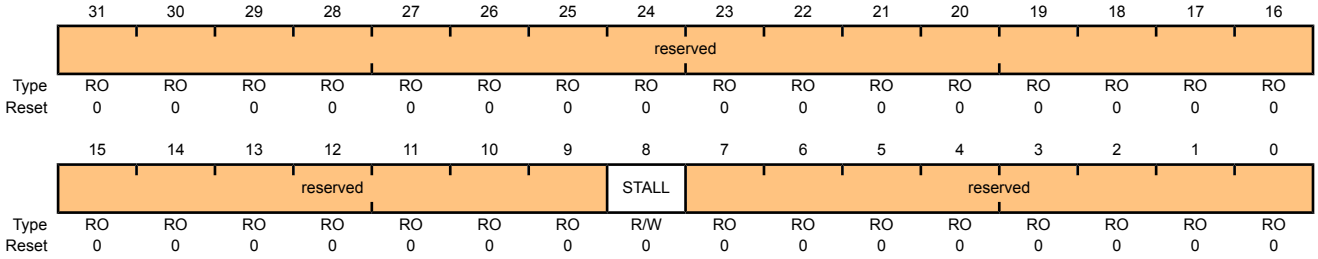
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	WDTMIS	RO	0	Watchdog Masked Interrupt Status Gives 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



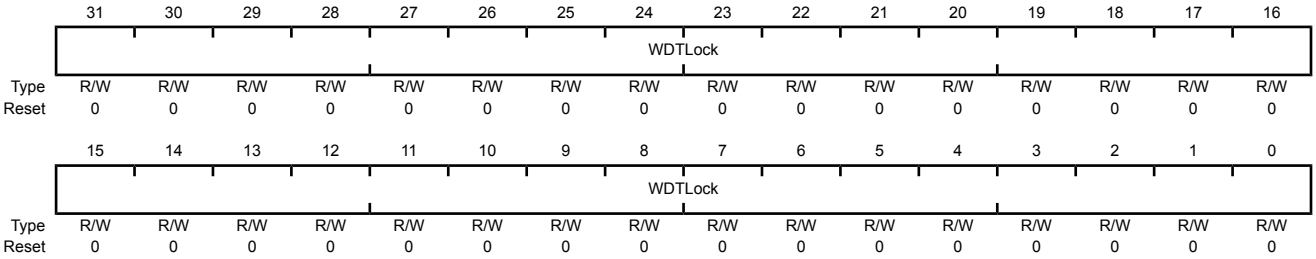
Bit/Field	Name	Type	Reset	Description
31:9	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	STALL	R/W	0	Watchdog Stall Enable When set to 1, 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



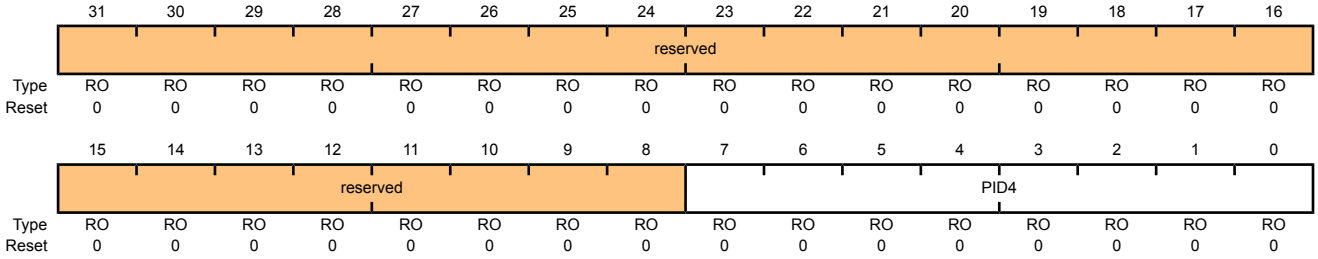
Bit/Field	Name	Type	Reset	Description
31:0	WDTLock	R/W	0x0000	Watchdog Lock A write of the value 0x1ACC.E551 unlocks the watchdog registers for write access. A write of any other value reapplies the lock, preventing any register updates. A read of this register returns the following values: Value Description 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



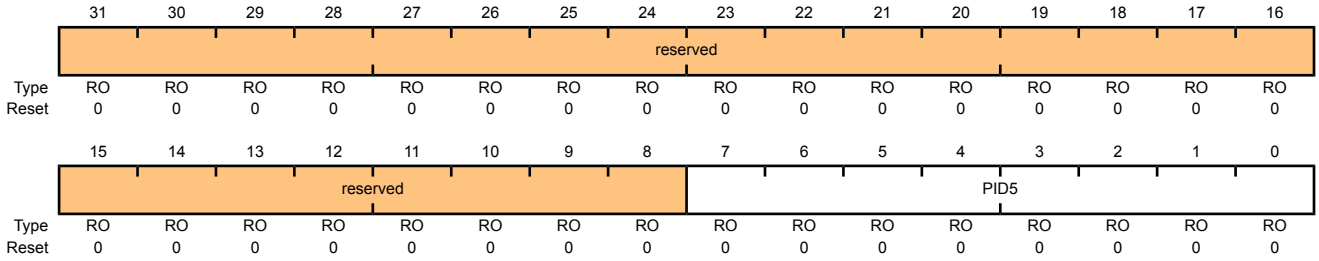
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	WDT Peripheral ID Register[7:0]

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



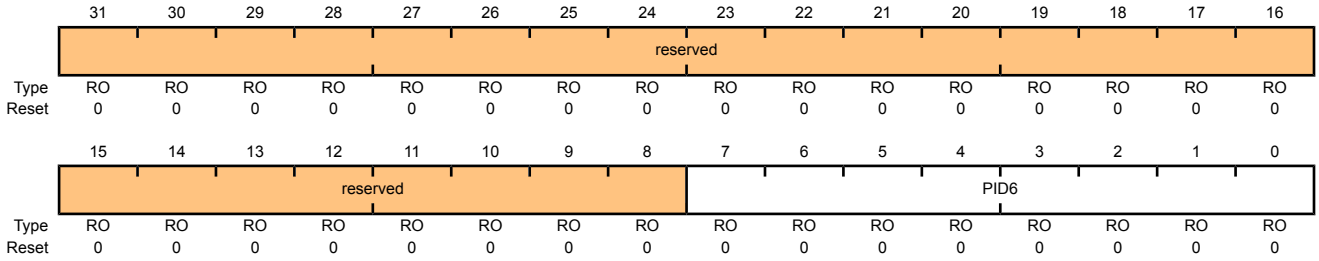
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	WDT Peripheral ID Register[15:8]

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



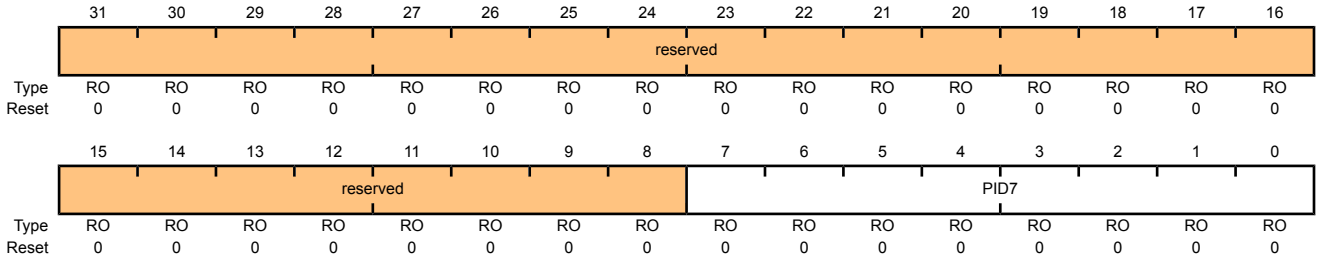
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	WDT Peripheral ID Register[23:16]

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



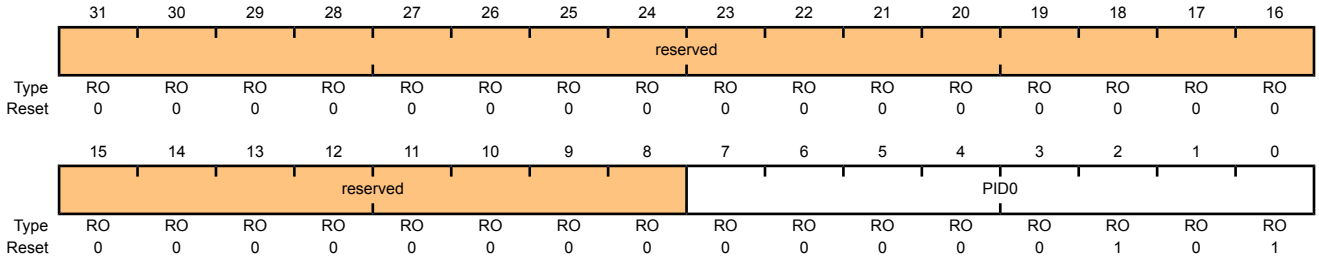
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	WDT Peripheral ID Register[31:24]

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



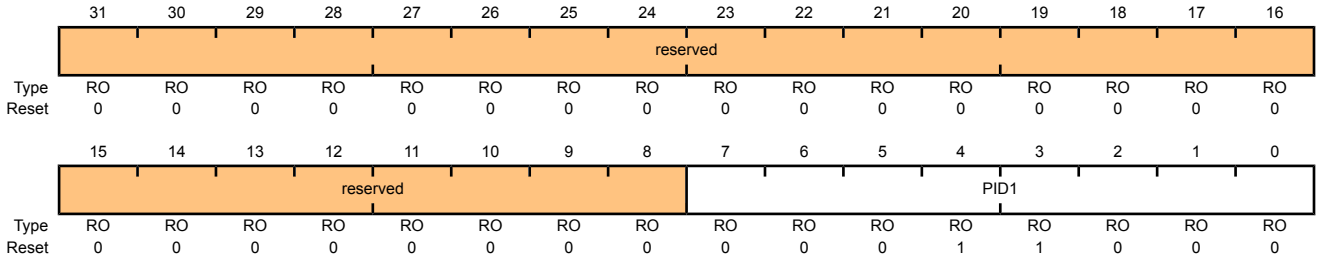
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x05	Watchdog Peripheral ID Register[7:0]

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



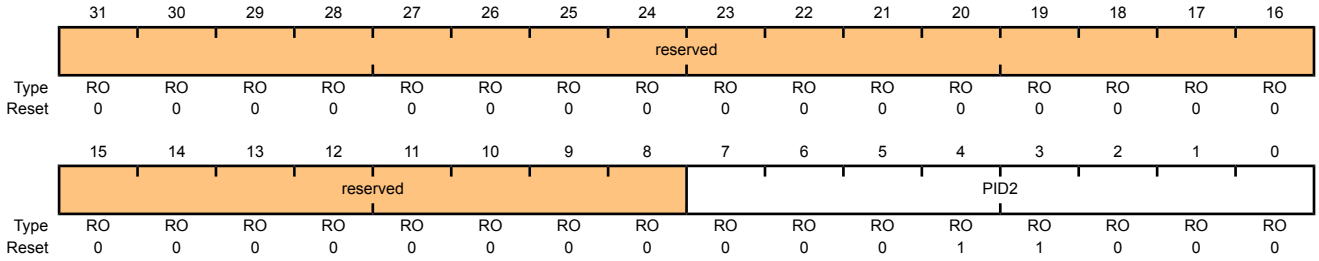
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x18	Watchdog Peripheral ID Register[15:8]

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



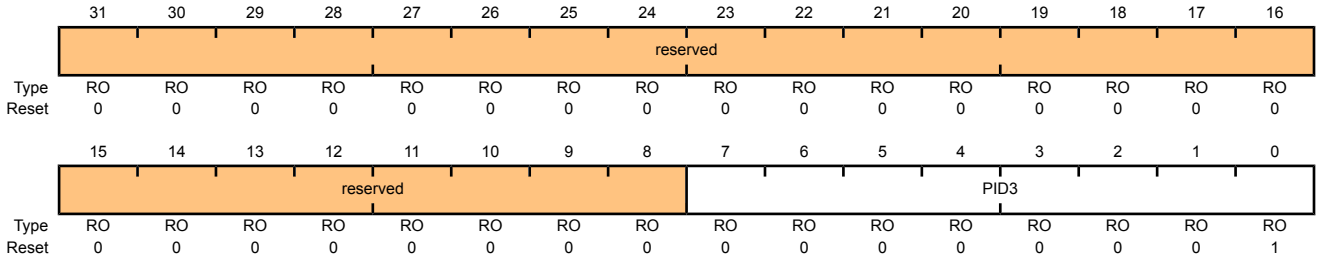
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	Watchdog Peripheral ID Register[23:16]

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



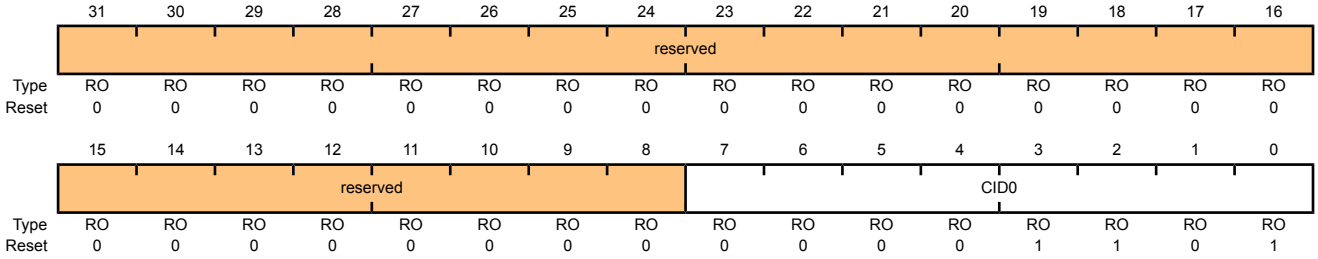
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	Watchdog Peripheral ID Register[31:24]

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



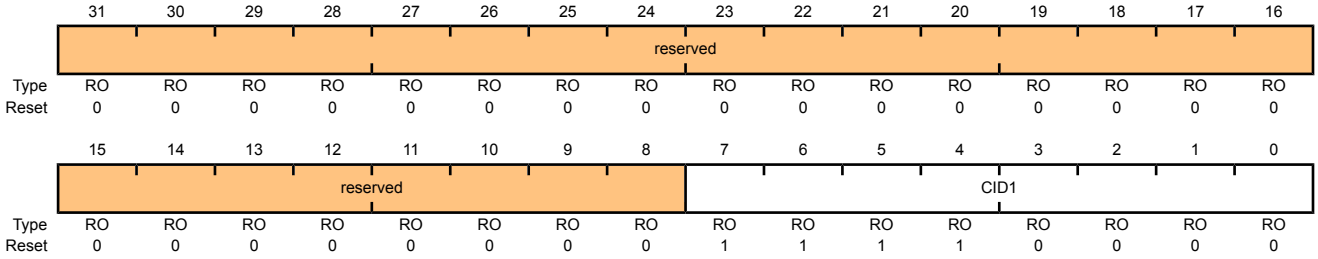
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	Watchdog PrimeCell ID Register[7:0]

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



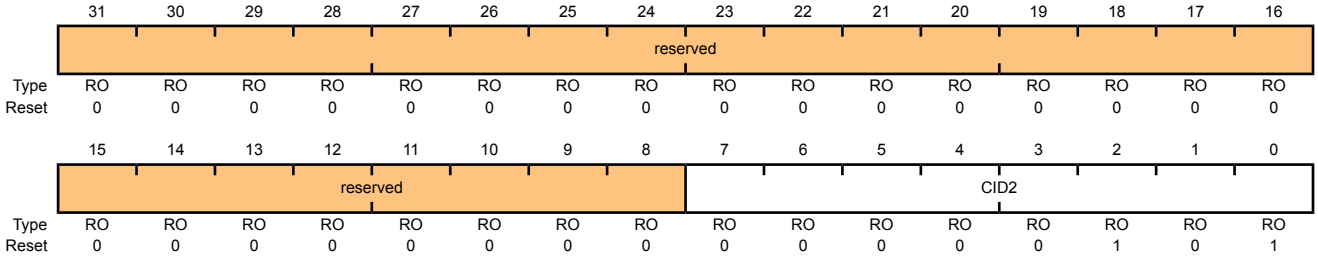
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	Watchdog PrimeCell ID Register[15:8]

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



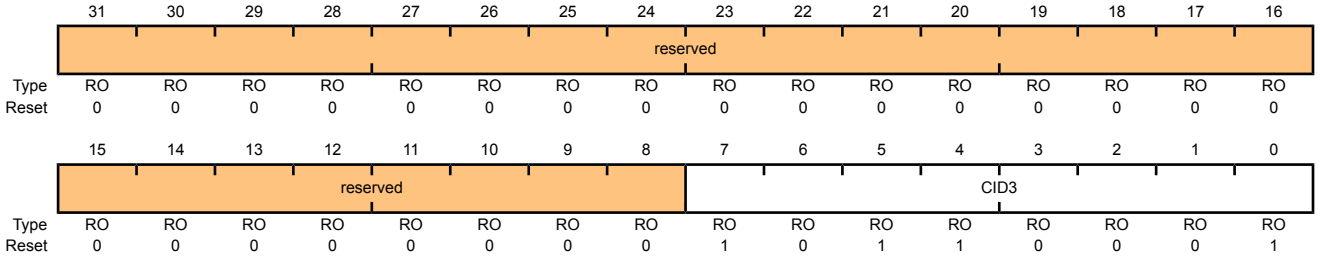
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	Watchdog PrimeCell ID Register[23:16]

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



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	Watchdog PrimeCell ID Register[31:24]

13 Analog-to-Digital Converter (ADC)

An analog-to-digital converter (ADC) is a peripheral that converts a continuous analog voltage to a discrete digital number.

The Stellaris® ADC module features 10-bit conversion resolution and supports eight input channels, plus an internal temperature sensor. The ADC module contains four programmable sequencer which allows for the sampling of multiple analog input sources without controller intervention. Each sample sequence provides flexible programming with fully configurable input source, trigger events, interrupt generation, and sequence priority.

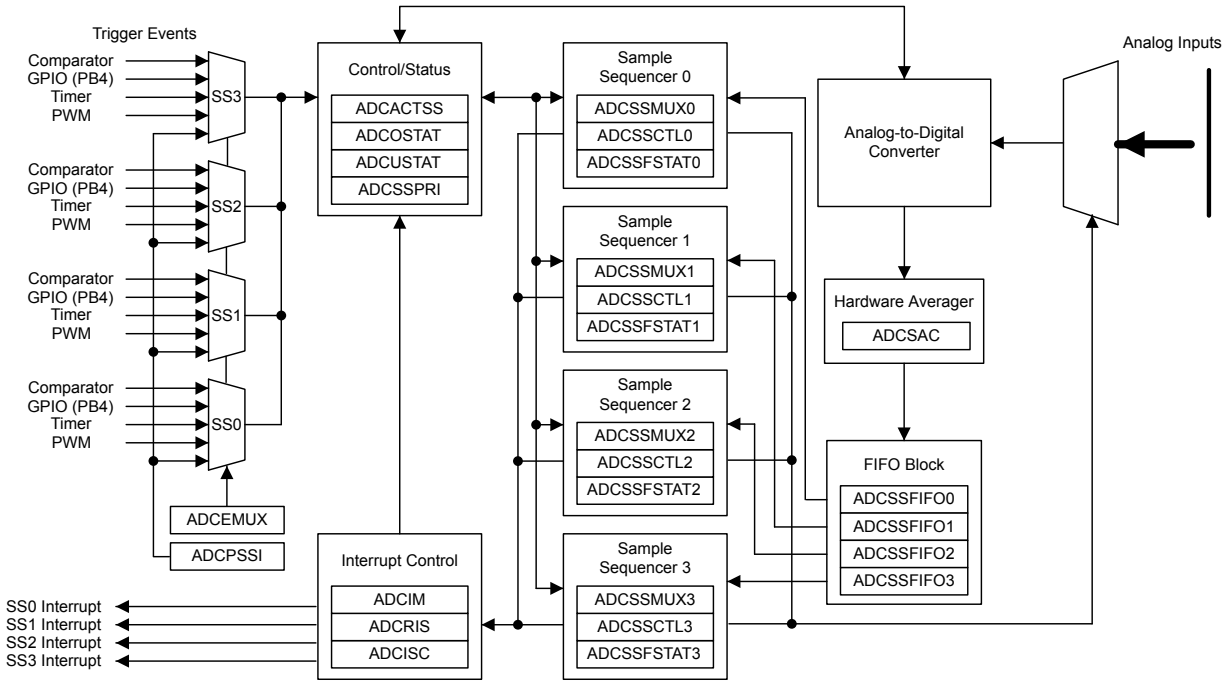
The Stellaris® ADC module provides the following features:

- Eight analog input channels
- Single-ended and differential-input configurations
- On-chip internal temperature sensor
- Sample rate of one million samples/second
- Flexible, configurable analog-to-digital conversion
- Four programmable sample conversion sequences from one to eight entries long, with corresponding conversion result FIFOs
- Flexible trigger control
  - Controller (software)
  - Timers
  - Analog Comparators
  - PWM
  - GPIO
- Hardware averaging of up to 64 samples for improved accuracy
- Converter uses an internal 3-V reference
- Power and ground for the analog circuitry is separate from the digital power and ground

13.1 Block Diagram

Figure 13-1 on page 362 provides details on the internal configuration of the ADC controls and data registers.

Figure 13-1. ADC Module Block Diagram



13.2 Functional Description

The Stellaris® ADC collects sample data by using a programmable sequence-based approach instead of the traditional single or double-sampling approaches found on many ADC modules. Each *sample sequence* is a fully programmed series of consecutive (back-to-back) samples, allowing the ADC to collect data from multiple input sources without having to be re-configured or serviced by the controller. The programming of each sample in the sample sequence includes parameters such as the input source and mode (differential versus single-ended input), interrupt generation on sample completion, and the indicator for the last sample in the sequence.

13.2.1 Sample Sequencers

The sampling control and data capture is handled by the sample sequencers. All of the sequencers are identical in implementation except for the number of samples that can be captured and the depth of the FIFO. Table 13-1 on page 362 shows the maximum number of samples that each sequencer can capture and its corresponding FIFO depth. In this implementation, each FIFO entry is a 32-bit word, with the lower 10 bits containing the conversion result.

Table 13-1. Samples and FIFO Depth of Sequencers

Sequencer	Number of Samples	Depth of FIFO
SS3	1	1
SS2	4	4
SS1	4	4
SS0	8	8

For a given sample sequence, each sample is defined by two 4-bit nibbles in the **ADC Sample Sequence Input Multiplexer Select (ADCSSMUXn)** and **ADC Sample Sequence Control**

(ADCSSCTLn) registers, where "n" corresponds to the sequence number. The **ADCSSMUXn** nibbles select the input pin, while the **ADCSSCTLn** nibbles contain the sample control bits corresponding to parameters such as temperature sensor selection, interrupt enable, end of sequence, and differential input mode. Sample sequencers are enabled by setting the respective ASENn bit in the **ADC Active Sample Sequencer (ADCACTSS)** register, and should be configured before being enabled.

When configuring a sample sequence, multiple uses of the same input pin within the same sequence is allowed. In the **ADCSSCTLn** register, the IEn bits can be set for any combination of samples, allowing interrupts to be generated after every sample in the sequence if necessary. Also, the END bit can be set at any point within a sample sequence. For example, if Sequencer 0 is used, the END bit can be set in the nibble associated with the fifth sample, allowing Sequencer 0 to complete execution of the sample sequence after the fifth sample.

After a sample sequence completes execution, the result data can be retrieved from the **ADC Sample Sequence Result FIFO (ADCSSFIFOn)** registers. The FIFOs are simple circular buffers that read a single address to "pop" result data. For software debug purposes, the positions of the FIFO head and tail pointers are visible in the **ADC Sample Sequence FIFO Status (ADCSSFSTATn)** registers along with FULL and EMPTY status flags. Overflow and underflow conditions are monitored using the **ADCOSTAT** and **ADCUSTAT** registers.

13.2.2 Module Control

Outside of the sample sequencers, the remainder of the control logic is responsible for tasks such as:

- Interrupt generation
- Sequence prioritization
- Trigger configuration

Most of the ADC control logic runs at the ADC clock rate of 14-18 MHz. The internal ADC divider is configured automatically by hardware when the system XTAL is selected. The automatic clock divider configuration targets 16.667 MHz operation for all Stellaris® devices.

13.2.2.1 Interrupts

The register configurations of the sample sequencers dictate which events generate raw interrupts, but do not have control over whether the interrupt is actually sent to the interrupt controller. The ADC module's interrupt signals are controlled by the state of the MASK bits in the **ADC Interrupt Mask (ADCIM)** register. Interrupt status can be viewed at two locations: the **ADC Raw Interrupt Status (ADCRIS)** register, which shows the raw status of the various interrupt signals, and the **ADC Interrupt Status and Clear (ADCISC)** register, which shows active interrupts that are enabled by the **ADCIM** register. Sequencer interrupts are cleared by writing a 1 to the corresponding IN bit in **ADCISC**.

13.2.2.2 Prioritization

When sampling events (triggers) happen concurrently, they are prioritized for processing by the values in the **ADC Sample Sequencer Priority (ADCSSPRI)** register. Valid priority values are in the range of 0-3, with 0 being the highest priority and 3 being the lowest. Multiple active sample sequencer units with the same priority do not provide consistent results, so software must ensure that all active sample sequencer units have a unique priority value.

13.2.2.3 Sampling Events

Sample triggering for each sample sequencer is defined in the **ADC Event Multiplexer Select (ADCEMUX)** register. The external peripheral triggering sources vary by Stellaris® family member, but all devices share the "Controller" and "Always" triggers. Software can initiate sampling by setting the SSx bits in the **ADC Processor Sample Sequence Initiate (ADCPSSI)** register.

Care must be taken when using the "Always" trigger. If a sequence's priority is too high, it is possible to starve other lower priority sequences.

13.2.3 Hardware Sample Averaging Circuit

Higher precision results can be generated using the hardware averaging circuit, however, the improved results are at the cost of throughput. Up to 64 samples can be accumulated and averaged to form a single data entry in the sequencer FIFO. Throughput is decreased proportionally to the number of samples in the averaging calculation. For example, if the averaging circuit is configured to average 16 samples, the throughput is decreased by a factor of 16.

By default the averaging circuit is off and all data from the converter passes through to the sequencer FIFO. The averaging hardware is controlled by the **ADC Sample Averaging Control (ADCSAC)** register (see page 383). There is a single averaging circuit and all input channels receive the same amount of averaging whether they are single-ended or differential.

13.2.4 Analog-to-Digital Converter

The converter itself generates a 10-bit output value for selected analog input. Special analog pads are used to minimize the distortion on the input. An internal 3 V reference is used by the converter resulting in sample values ranging from 0x000 at 0 V input to 0x3FF at 3 V input when in single-ended input mode.

13.2.5 Differential Sampling

In addition to traditional single-ended sampling, the ADC module supports differential sampling of two analog input channels. To enable differential sampling, software must set the Dn bit in the **ADCSSCTL0n** register in a step's configuration nibble.

When a sequence step is configured for differential sampling, its corresponding value in the **ADCSSMUXn** register must be set to one of the four differential pairs, numbered 0-3. Differential pair 0 samples analog inputs 0 and 1; differential pair 1 samples analog inputs 2 and 3; and so on (see Table 13-2 on page 364). The ADC does not support other differential pairings such as analog input 0 with analog input 3. The number of differential pairs supported is dependent on the number of analog inputs (see Table 13-2 on page 364).

Table 13-2. Differential Sampling Pairs

Differential Pair	Analog Inputs
0	0 and 1
1	2 and 3
2	4 and 5
3	6 and 7

The voltage sampled in differential mode is the difference between the odd and even channels:

ΔV (differential voltage) = V_{IN_EVEN} (even channels) – V_{IN_ODD} (odd channels), therefore:

- If $\Delta V = 0$, then the conversion result = 0x1FF

- If ΔV > 0, then the conversion result > 0x1FF (range is 0x1FF–0x3FF)
- If ΔV < 0, then the conversion result < 0x1FF (range is 0–0x1FF)

The differential pairs assign polarities to the analog inputs: the even-numbered input is always positive, and the odd-numbered input is always negative. In order for a valid conversion result to appear, the negative input must be in the range of ± 1.5 V of the positive input. If an analog input is greater than 3 V or less than 0 V (the valid range for analog inputs), the input voltage is clipped, meaning it appears as either 3 V or 0 V, respectively, to the ADC.

Figure 13-2 on page 365 shows an example of the negative input centered at 1.5 V. In this configuration, the differential range spans from -1.5 V to 1.5 V. Figure 13-3 on page 366 shows an example where the negative input is centered at -0.75 V, meaning inputs on the positive input saturate past a differential voltage of -0.75 V since the input voltage is less than 0 V. Figure 13-4 on page 366 shows an example of the negative input centered at 2.25 V, where inputs on the positive channel saturate past a differential voltage of 0.75 V since the input voltage would be greater than 3 V.

Figure 13-2. Differential Sampling Range, V_{IN_ODD} = 1.5 V

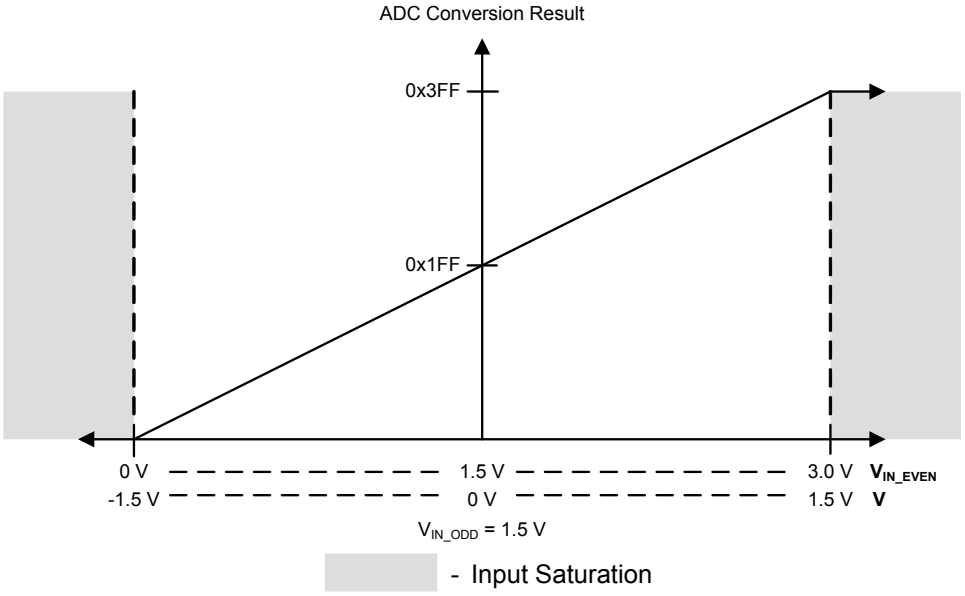


Figure 13-3. Differential Sampling Range, V_{IN_ODD} = 0.75 V

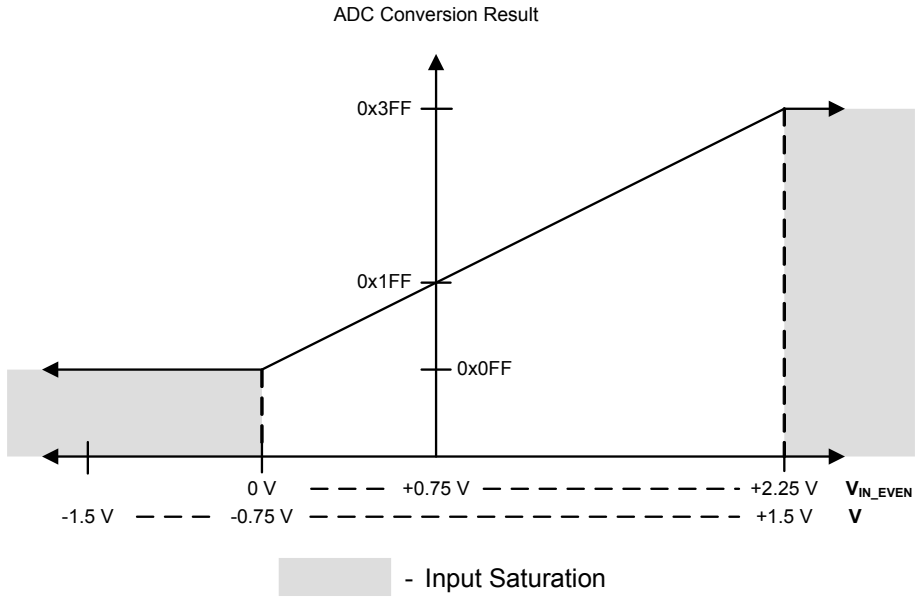
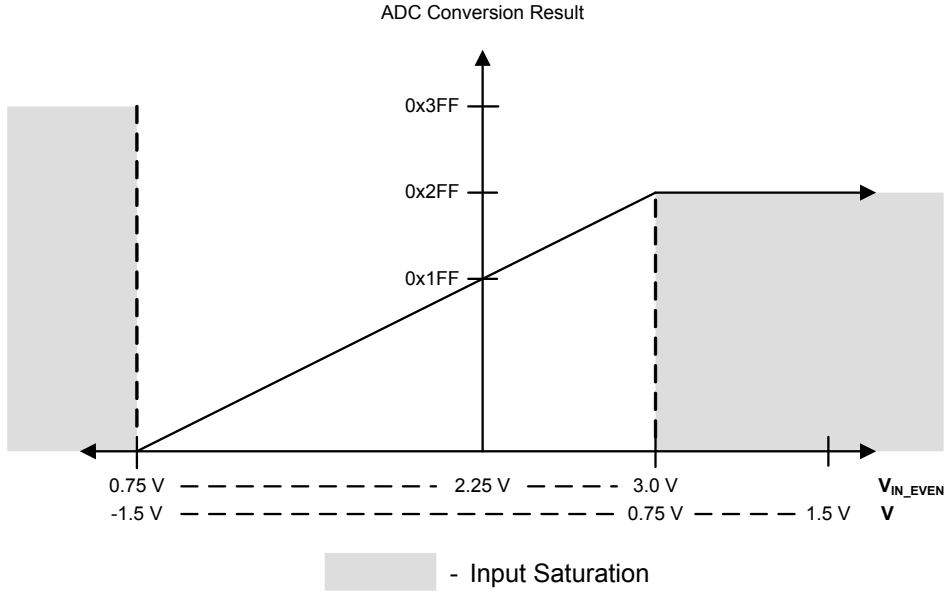


Figure 13-4. Differential Sampling Range, V_{IN_ODD} = 2.25 V



13.2.6 Internal Temperature Sensor

The temperature sensor serves two primary purposes: 1) to notify the system that internal temperature is too high or low for reliable operation, and 2) to provide temperature measurements for calibration of the Hibernate module RTC trim value.

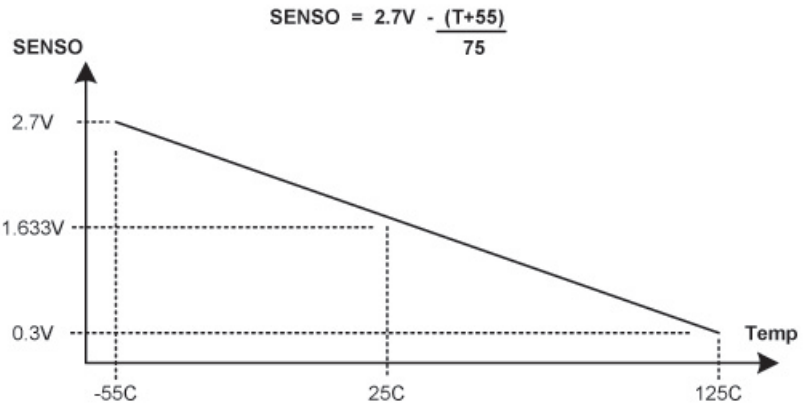
The temperature sensor does not have a separate enable, since it also contains the bandgap reference and must always be enabled. The reference is supplied to other analog modules; not just the ADC.

The internal temperature sensor provides an analog temperature reading as well as a reference voltage. The voltage at the output terminal SENSO is given by the following equation:

SENSO = 2.7 - ((T + 55) / 75)

This relation is shown in Figure 13-5 on page 367.

Figure 13-5. Internal Temperature Sensor Characteristic



13.3 Initialization and Configuration

In order for the ADC module to be used, the PLL must be enabled and using a supported crystal frequency (see the **RCC** register). Using unsupported frequencies can cause faulty operation in the ADC module.

13.3.1 Module Initialization

Initialization of the ADC module is a simple process with very few steps. The main steps include enabling the clock to the ADC, disabling the analog isolation circuit associated with all inputs that are to be used, and reconfiguring the sample sequencer priorities (if needed).

The initialization sequence for the ADC is as follows:

1. Enable the ADC clock by writing a value of 0x0001.0000 to the **RCGC0** register (see page 118).
2. Disable the analog isolation circuit for all ADC input pins that are to be used by writing a 1 to the appropriate bits of the **GPIOAMSEL** register (see page 289) in the associated GPIO block.
3. If required by the application, reconfigure the sample sequencer priorities in the **ADCSSPRI** register. The default configuration has Sample Sequencer 0 with the highest priority, and Sample Sequencer 3 as the lowest priority.

13.3.2 Sample Sequencer Configuration

Configuration of the sample sequencers is slightly more complex than the module initialization since each sample sequence is completely programmable.

The configuration for each sample sequencer should be as follows:

1. Ensure that the sample sequencer is disabled by writing a 0 to the corresponding ASENn bit in the **ADCACTSS** register. Programming of the sample sequencers is allowed without having them enabled. Disabling the sequencer during programming prevents erroneous execution if a trigger event were to occur during the configuration process.
2. Configure the trigger event for the sample sequencer in the **ADCEMUX** register.
3. For each sample in the sample sequence, configure the corresponding input source in the **ADCSSMUXn** register.
4. For each sample in the sample sequence, configure the sample control bits in the corresponding nibble in the **ADCSSCTLn** register. When programming the last nibble, ensure that the END bit is set. Failure to set the END bit causes unpredictable behavior.
5. If interrupts are to be used, write a 1 to the corresponding MASK bit in the **ADCIM** register.
6. Enable the sample sequencer logic by writing a 1 to the corresponding ASENn bit in the **ADCACTSS** register.

13.4 Register Map

Table 13-3 on page 368 lists the ADC registers. The offset listed is a hexadecimal increment to the register's address, relative to the ADC base address of 0x4003.8000.

Table 13-3. ADC Register Map

Offset	Name	Type	Reset	Description	See page
0x000	ADCACTSS	R/W	0x0000.0000	ADC Active Sample Sequencer	370
0x004	ADCRIS	RO	0x0000.0000	ADC Raw Interrupt Status	371
0x008	ADCIM	R/W	0x0000.0000	ADC Interrupt Mask	372
0x00C	ADCISC	R/W1C	0x0000.0000	ADC Interrupt Status and Clear	373
0x010	ADCOSTAT	R/W1C	0x0000.0000	ADC Overflow Status	375
0x014	ADCEMUX	R/W	0x0000.0000	ADC Event Multiplexer Select	376
0x018	ADCUSTAT	R/W1C	0x0000.0000	ADC Underflow Status	379
0x020	ADCSSPRI	R/W	0x0000.3210	ADC Sample Sequencer Priority	380
0x028	ADCPSSI	WO	-	ADC Processor Sample Sequence Initiate	382
0x030	ADCSAC	R/W	0x0000.0000	ADC Sample Averaging Control	383
0x040	ADCSSMUX0	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 0	384
0x044	ADCSSCTL0	R/W	0x0000.0000	ADC Sample Sequence Control 0	386
0x048	ADCSSFIFO0	RO	0x0000.0000	ADC Sample Sequence Result FIFO 0	389
0x04C	ADCSSFSTAT0	RO	0x0000.0100	ADC Sample Sequence FIFO 0 Status	390
0x060	ADCSSMUX1	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 1	391
0x064	ADCSSCTL1	R/W	0x0000.0000	ADC Sample Sequence Control 1	392
0x068	ADCSSFIFO1	RO	0x0000.0000	ADC Sample Sequence Result FIFO 1	389

Offset	Name	Type	Reset	Description	See page
0x06C	ADCSSFSTAT1	RO	0x0000.0100	ADC Sample Sequence FIFO 1 Status	390
0x080	ADCSSMUX2	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 2	391
0x084	ADCSSCTL2	R/W	0x0000.0000	ADC Sample Sequence Control 2	392
0x088	ADCSSFIFO2	RO	0x0000.0000	ADC Sample Sequence Result FIFO 2	389
0x08C	ADCSSFSTAT2	RO	0x0000.0100	ADC Sample Sequence FIFO 2 Status	390
0x0A0	ADCSSMUX3	R/W	0x0000.0000	ADC Sample Sequence Input Multiplexer Select 3	394
0x0A4	ADCSSCTL3	R/W	0x0000.0002	ADC Sample Sequence Control 3	395
0x0A8	ADCSSFIFO3	RO	0x0000.0000	ADC Sample Sequence Result FIFO 3	389
0x0AC	ADCSSFSTAT3	RO	0x0000.0100	ADC Sample Sequence FIFO 3 Status	390

13.5 Register Descriptions

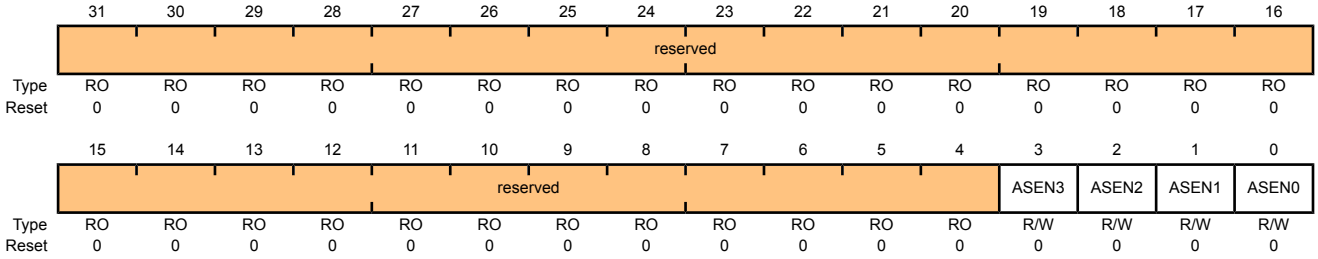
The remainder of this section lists and describes the ADC registers, in numerical order by address offset.

Register 1: ADC Active Sample Sequencer (ADCACTSS), offset 0x000

This register controls the activation of the sample sequencers. Each sample sequencer can be enabled or disabled independently.

ADC Active Sample Sequencer (ADCACTSS)

Base 0x4003.8000
Offset 0x000
Type R/W, reset 0x0000.0000



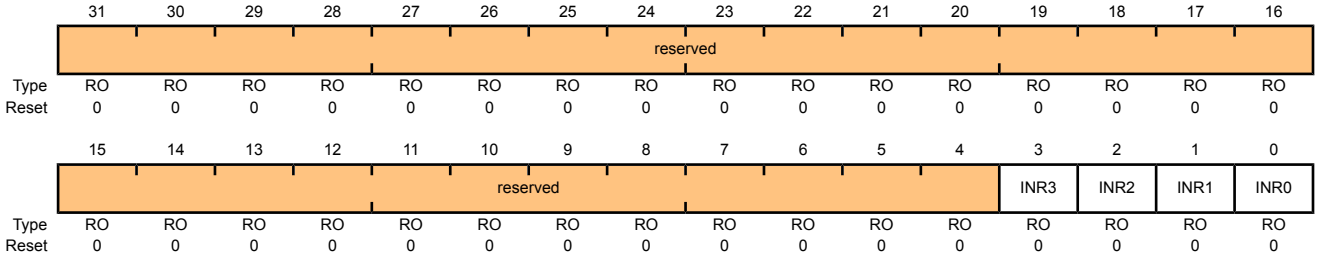
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ASEN3	R/W	0	ADC SS3 Enable Specifies whether Sample Sequencer 3 is enabled. If set, the sample sequence logic for Sequencer 3 is active. Otherwise, the sequencer is inactive.
2	ASEN2	R/W	0	ADC SS2 Enable Specifies whether Sample Sequencer 2 is enabled. If set, the sample sequence logic for Sequencer 2 is active. Otherwise, the sequencer is inactive.
1	ASEN1	R/W	0	ADC SS1 Enable Specifies whether Sample Sequencer 1 is enabled. If set, the sample sequence logic for Sequencer 1 is active. Otherwise, the sequencer is inactive.
0	ASEN0	R/W	0	ADC SS0 Enable Specifies whether Sample Sequencer 0 is enabled. If set, the sample sequence logic for Sequencer 0 is active. Otherwise, the sequencer is inactive.

Register 2: ADC Raw Interrupt Status (ADCRIS), offset 0x004

This register shows the status of the raw interrupt signal of each sample sequencer. These bits may be polled by software to look for interrupt conditions without having to generate controller interrupts.

ADC Raw Interrupt Status (ADCRIS)

Base 0x4003.8000
Offset 0x004
Type RO, reset 0x0000.0000



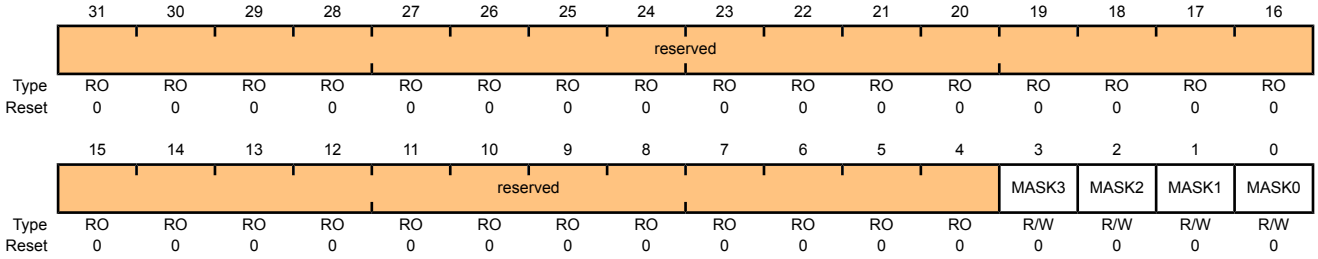
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	INR3	RO	0	SS3 Raw Interrupt Status This bit is set by hardware when a sample with its respective **ADCSSCTL3** `IE` bit has completed conversion. This bit is cleared by setting the `IN3` bit in the **ADCISC** register.
2	INR2	RO	0	SS2 Raw Interrupt Status This bit is set by hardware when a sample with its respective **ADCSSCTL2** `IE` bit has completed conversion. This bit is cleared by setting the `IN2` bit in the **ADCISC** register.
1	INR1	RO	0	SS1 Raw Interrupt Status This bit is set by hardware when a sample with its respective **ADCSSCTL1** `IE` bit has completed conversion. This bit is cleared by setting the `IN1` bit in the **ADCISC** register.
0	INR0	RO	0	SS0 Raw Interrupt Status This bit is set by hardware when a sample with its respective **ADCSSCTL0** `IE` bit has completed conversion. This bit is cleared by setting the `IN30` bit in the **ADCISC** register.

Register 3: ADC Interrupt Mask (ADCIM), offset 0x008

This register controls whether the sample sequencer raw interrupt signals are promoted to controller interrupts. Each raw interrupt signal can be masked independently.

ADC Interrupt Mask (ADCIM)

Base 0x4003.8000
Offset 0x008
Type R/W, reset 0x0000.0000



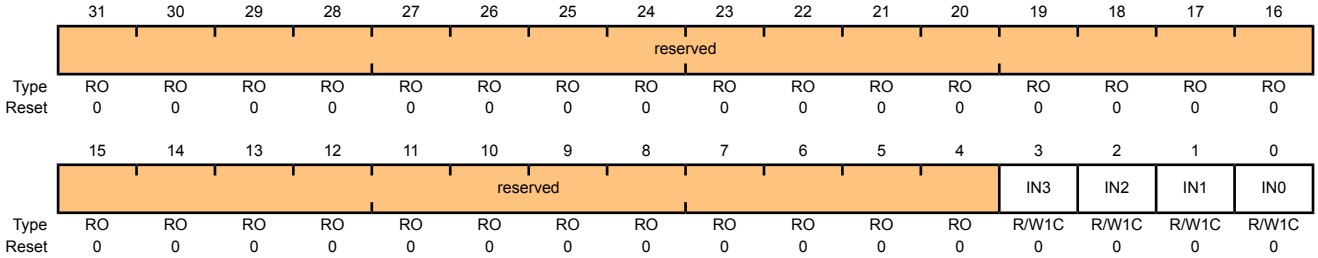
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	MASK3	R/W	0	SS3 Interrupt Mask When set, this bit allows the raw interrupt signal from Sample Sequencer 3 (**ADCRIS** register `INR3` bit) to be promoted to a controller interrupt. When clear, the status of Sample Sequencer 3 does not affect the SS3 interrupt status.
2	MASK2	R/W	0	SS2 Interrupt Mask When set, this bit allows the raw interrupt signal from Sample Sequencer 2 (**ADCRIS** register `INR2` bit) to be promoted to a controller interrupt. When clear, the status of Sample Sequencer 2 does not affect the SS2 interrupt status.
1	MASK1	R/W	0	SS1 Interrupt Mask When set, this bit allows the raw interrupt signal from Sample Sequencer 1 (**ADCRIS** register `INR1` bit) to be promoted to a controller interrupt. When clear, the status of Sample Sequencer 1 does not affect the SS1 interrupt status.
0	MASK0	R/W	0	SS0 Interrupt Mask When set, this bit allows the raw interrupt signal from Sample Sequencer 0 (**ADCRIS** register `INR0` bit) to be promoted to a controller interrupt. When clear, the status of Sample Sequencer 0 does not affect the SS0 interrupt status.

Register 4: ADC Interrupt Status and Clear (ADCISC), offset 0x00C

This register provides the mechanism for clearing sample sequence interrupt conditions and shows the status of controller interrupts generated by the sample sequencers. When read, each bit field is the logical AND of the respective INR and MASK bits. Sample sequence nterrupts are cleared by setting the corresponding bit position. If software is polling the **ADCRIS** instead of generating interrupts, the sample sequence INR bits are still cleared via the **ADCISC** register, even if the IN bit is not set.

ADC Interrupt Status and Clear (ADCISC)

Base 0x4003.8000
Offset 0x00C
Type R/W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IN3	R/W1C	0	SS3 Interrupt Status and Clear This bit is set when both the INR3 bit in the **ADCRIS** register and the MASK3 bit in the **ADCIM** register are set, providing a level-based interrupt to the controller. This bit is cleared by writing a 1. Clearing this bit also clears the INR3 bit.
2	IN2	R/W1C	0	SS2 Interrupt Status and Clear This bit is set when both the INR2 bit in the **ADCRIS** register and the MASK2 bit in the **ADCIM** register are set, providing a level-based interrupt to the controller. This bit is cleared by writing a 1. Clearing this bit also clears the INR2 bit.
1	IN1	R/W1C	0	SS1 Interrupt Status and Clear This bit is set when both the INR1 bit in the **ADCRIS** register and the MASK1 bit in the **ADCIM** register are set, providing a level-based interrupt to the controller. This bit is cleared by writing a 1. Clearing this bit also clears the INR1 bit.

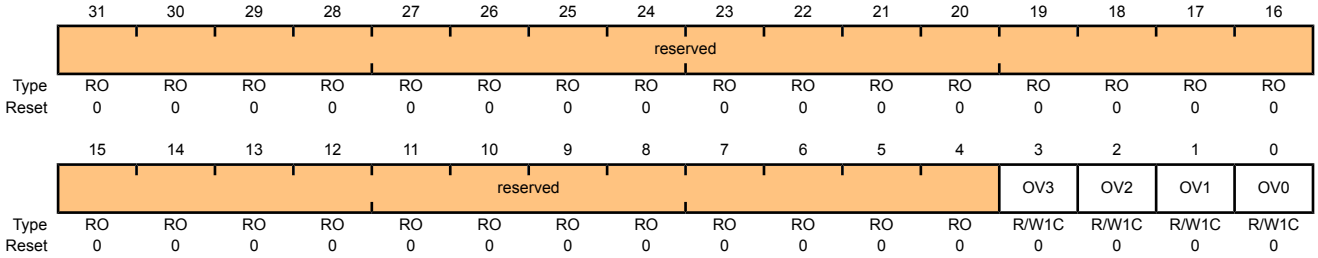
Bit/Field	Name	Type	Reset	Description
0	IN0	R/W1C	0	SS0 Interrupt Status and Clear This bit is set when both the INR0 bit in the **ADCRIS** register and the MASK0 bit in the **ADCIM** register are set, providing a level-based interrupt to the controller. This bit is cleared by writing a 1. Clearing this bit also clears the INR0 bit.

Register 5: ADC Overflow Status (ADCOSTAT), offset 0x010

This register indicates overflow conditions in the sample sequencer FIFOs. Once the overflow condition has been handled by software, the condition can be cleared by writing a 1 to the corresponding bit position.

ADC Overflow Status (ADCOSTAT)

Base 0x4003.8000
Offset 0x010
Type R/W1C, reset 0x0000.0000



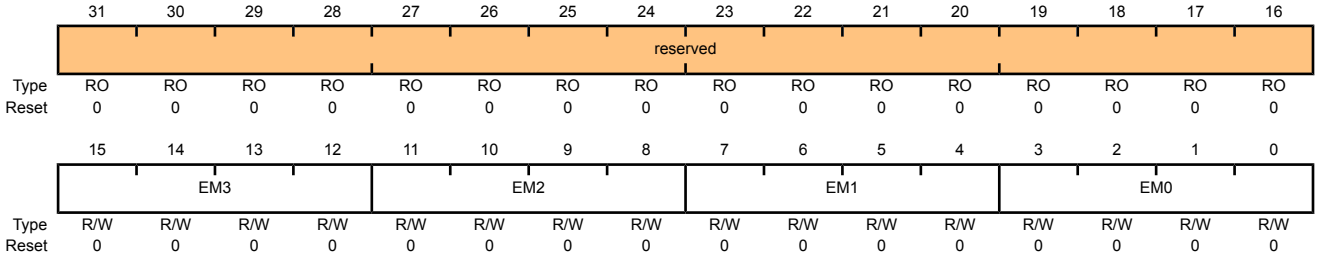
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OV3	R/W1C	0	SS3 FIFO Overflow When set, this bit specifies that the FIFO for Sample Sequencer 3 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. This bit is cleared by writing a 1.
2	OV2	R/W1C	0	SS2 FIFO Overflow When set, this bit specifies that the FIFO for Sample Sequencer 2 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. This bit is cleared by writing a 1.
1	OV1	R/W1C	0	SS1 FIFO Overflow When set, this bit specifies that the FIFO for Sample Sequencer 1 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. This bit is cleared by writing a 1.
0	OV0	R/W1C	0	SS0 FIFO Overflow When set, this bit specifies that the FIFO for Sample Sequencer 0 has hit an overflow condition where the FIFO is full and a write was requested. When an overflow is detected, the most recent write is dropped. This bit is cleared by writing a 1.

Register 6: ADC Event Multiplexer Select (ADCEMUX), offset 0x014

The **ADCEMUX** selects the event (trigger) that initiates sampling for each sample sequencer. Each sample sequencer can be configured with a unique trigger source.

ADC Event Multiplexer Select (ADCEMUX)

Base 0x4003.8000
Offset 0x014
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:12	EM3	R/W	0x0	SS3 Trigger Select This field selects the trigger source for Sample Sequencer 3. The valid configurations for this field are: Value Event 0x0 Controller (default) 0x1 Analog Comparator 0 0x2 Analog Comparator 1 0x3 Reserved 0x4 External (GPIO PB4) 0x5 Timer In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (see page 320). 0x6 PWM0 0x7 PWM1 0x8 PWM2 0x9-0xE reserved 0xF Always (continuously sample)

Bit/Field	Name	Type	Reset	Description
11:8	EM2	R/W	0x0	SS2 Trigger Select This field selects the trigger source for Sample Sequencer 2. The valid configurations for this field are: Value Event 0x0 Controller (default) 0x1 Analog Comparator 0 0x2 Analog Comparator 1 0x3 Reserved 0x4 External (GPIO PB4) 0x5 Timer In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (see page 320). 0x6 PWM0 0x7 PWM1 0x8 PWM2 0x9-0xE reserved 0xF Always (continuously sample)
7:4	EM1	R/W	0x0	SS1 Trigger Select This field selects the trigger source for Sample Sequencer 1. The valid configurations for this field are: Value Event 0x0 Controller (default) 0x1 Analog Comparator 0 0x2 Analog Comparator 1 0x3 Reserved 0x4 External (GPIO PB4) 0x5 Timer In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (see page 320). 0x6 PWM0 0x7 PWM1 0x8 PWM2 0x9-0xE reserved 0xF Always (continuously sample)

Bit/Field	Name	Type	Reset	Description
3:0	EM0	R/W	0x0	SS0 Trigger Select This field selects the trigger source for Sample Sequencer 0. The valid configurations for this field are:

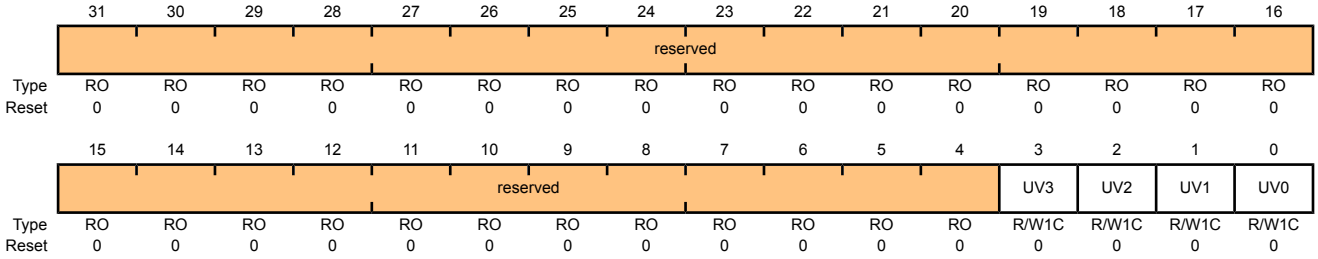
Value	Event
0x0	Controller (default)
0x1	Analog Comparator 0
0x2	Analog Comparator 1
0x3	Reserved
0x4	External (GPIO PB4)
0x5	Timer In addition, the trigger must be enabled with the TnOTE bit in the **GPTMCTL** register (see page 320).
0x6	PWM0
0x7	PWM1
0x8	PWM2
0x9-0xE	reserved
0xF	Always (continuously sample)

Register 7: ADC Underflow Status (ADCUSTAT), offset 0x018

This register indicates underflow conditions in the sample sequencer FIFOs. The corresponding underflow condition is cleared by writing a 1 to the relevant bit position.

ADC Underflow Status (ADCUSTAT)

Base 0x4003.8000
Offset 0x018
Type R/W1C, reset 0x0000.0000



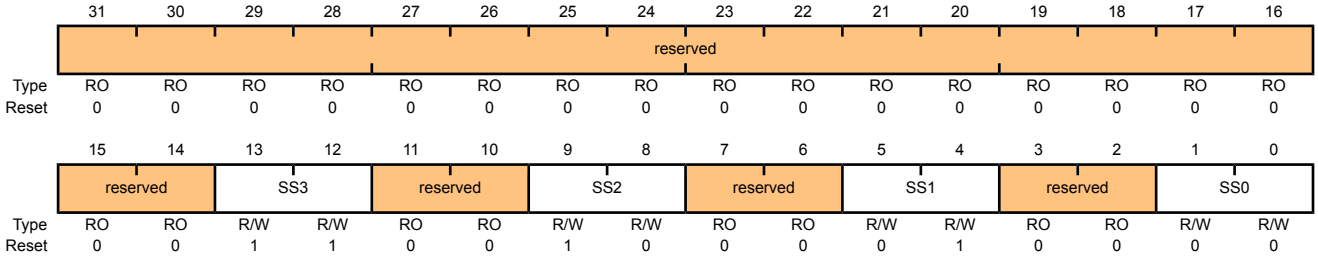
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	UV3	R/W1C	0	SS3 FIFO Underflow When set, this bit specifies that the FIFO for Sample Sequencer 3 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.
2	UV2	R/W1C	0	SS2 FIFO Underflow When set, this bit specifies that the FIFO for Sample Sequencer 2 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.
1	UV1	R/W1C	0	SS1 FIFO Underflow When set, this bit specifies that the FIFO for Sample Sequencer 1 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.
0	UV0	R/W1C	0	SS0 FIFO Underflow When set, this bit specifies that the FIFO for Sample Sequencer 0 has hit an underflow condition where the FIFO is empty and a read was requested. The problematic read does not move the FIFO pointers, and 0s are returned. This bit is cleared by writing a 1.

Register 8: ADC Sample Sequencer Priority (ADCSSPRI), offset 0x020

This register sets the priority for each of the sample sequencers. Out of reset, Sequencer 0 has the highest priority, and Sequencer 3 has the lowest priority. When reconfiguring sequence priorities, each sequence must have a unique priority for the ADC to operate properly.

ADC Sample Sequencer Priority (ADCSSPRI)

Base 0x4003.8000
Offset 0x020
Type R/W, reset 0x0000.3210



Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13:12	SS3	R/W	0x3	SS3 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 3. A priority encoding of 0 is highest and 3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
11:10	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	SS2	R/W	0x2	SS2 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 2. A priority encoding of 0 is highest and 3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
7:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:4	SS1	R/W	0x1	SS1 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 1. A priority encoding of 0 is highest and 3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.
3:2	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

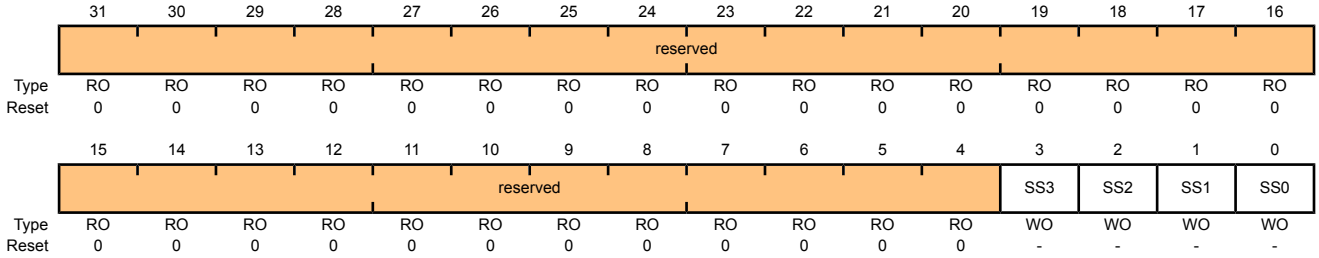
Bit/Field	Name	Type	Reset	Description
1:0	SS0	R/W	0x0	SS0 Priority This field contains a binary-encoded value that specifies the priority encoding of Sample Sequencer 0. A priority encoding of 0 is highest and 3 is lowest. The priorities assigned to the sequencers must be uniquely mapped. The ADC may not operate properly if two or more fields are equal.

Register 9: ADC Processor Sample Sequence Initiate (ADCPSSI), offset 0x028

This register provides a mechanism for application software to initiate sampling in the sample sequencers. Sample sequences can be initiated individually or in any combination. When multiple sequences are triggered simultaneously, the priority encodings in **ADCSSPRI** dictate execution order.

ADC Processor Sample Sequence Initiate (ADCPSSI)

Base 0x4003.8000
Offset 0x028
Type WO, reset -



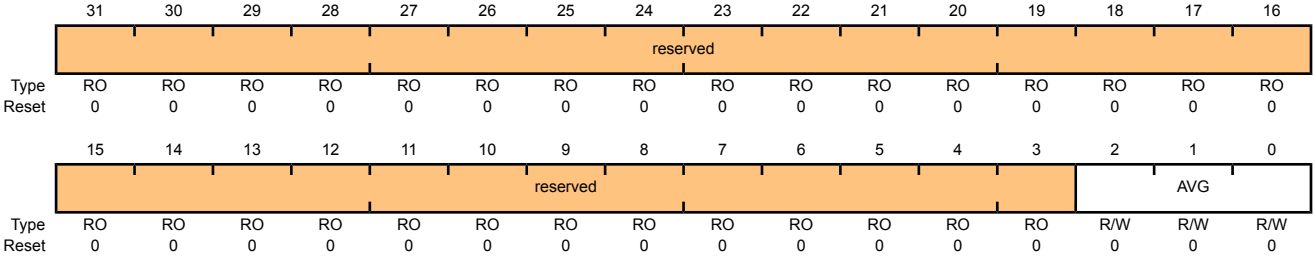
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	SS3	WO	-	SS3 Initiate When set, this bit triggers sampling on Sample Sequencer 3 if the sequencer is enabled in the **ADCACTSS** register. Only a write by software is valid; a read of this register returns no meaningful data.
2	SS2	WO	-	SS2 Initiate When set, this bit triggers sampling on Sample Sequencer 2 if the sequencer is enabled in the **ADCACTSS** register. Only a write by software is valid; a read of this register returns no meaningful data.
1	SS1	WO	-	SS1 Initiate When set, this bit triggers sampling on Sample Sequencer 1 if the sequencer is enabled in the **ADCACTSS** register. Only a write by software is valid; a read of this register returns no meaningful data.
0	SS0	WO	-	SS0 Initiate When set, this bit triggers sampling on Sample Sequencer 0 if the sequencer is enabled in the **ADCACTSS** register. Only a write by software is valid; a read of this register returns no meaningful data.

Register 10: ADC Sample Averaging Control (ADCSAC), offset 0x030

This register controls the amount of hardware averaging applied to conversion results. The final conversion result stored in the FIFO is averaged from 2^{AVG} consecutive ADC samples at the specified ADC speed. If AVG is 0, the sample is passed directly through without any averaging. If AVG=6, then 64 consecutive ADC samples are averaged to generate one result in the sequencer FIFO. An AVG = 7 provides unpredictable results.

ADC Sample Averaging Control (ADCSAC)

Base 0x4003.8000
Offset 0x030
Type R/W, reset 0x0000.0000



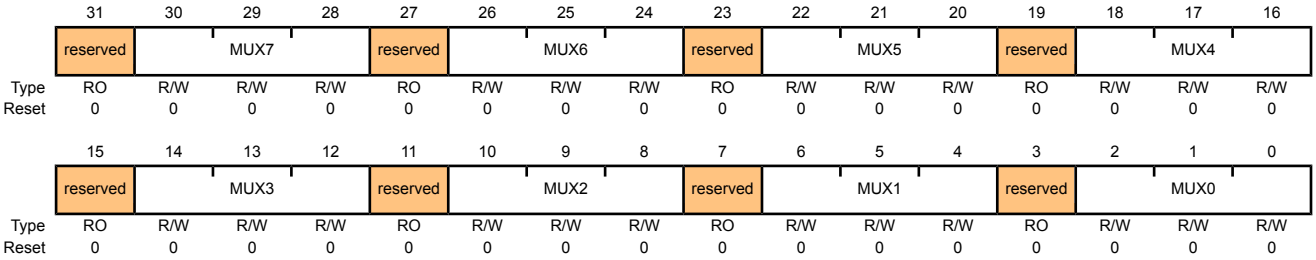
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	AVG	R/W	0x0	Hardware Averaging Control Specifies the amount of hardware averaging that will be applied to ADC samples. The AVG field can be any value between 0 and 6. Entering a value of 7 creates unpredictable results. Value Description 0x0 No hardware oversampling 0x1 2x hardware oversampling 0x2 4x hardware oversampling 0x3 8x hardware oversampling 0x4 16x hardware oversampling 0x5 32x hardware oversampling 0x6 64x hardware oversampling 0x7 Reserved

Register 11: ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0), offset 0x040

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 0. This register is 32 bits wide and contains information for eight possible samples.

ADC Sample Sequence Input Multiplexer Select 0 (ADCSSMUX0)

Base 0x4003.8000
Offset 0x040
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
30:28	MUX7	R/W	0x0	8th Sample Input Select The MUX7 field is used during the eighth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion. The value set here indicates the corresponding pin, for example, a value of 1 indicates the input is ADC1.
27	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
26:24	MUX6	R/W	0x0	7th Sample Input Select The MUX6 field is used during the seventh sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
23	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
22:20	MUX5	R/W	0x0	6th Sample Input Select The MUX5 field is used during the sixth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
19	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
18:16	MUX4	R/W	0x0	5th Sample Input Select The MUX4 field is used during the fifth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
15	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14:12	MUX3	R/W	0x0	4th Sample Input Select The MUX3 field is used during the fourth sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10:8	MUX2	R/W	0x0	3rd Sample Input Select The MUX72 field is used during the third sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:4	MUX1	R/W	0x0	2nd Sample Input Select The MUX1 field is used during the second sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	MUX0	R/W	0x0	1st Sample Input Select The MUX0 field is used during the first sample of a sequence executed with the sample sequencer. It specifies which of the analog inputs is sampled for the analog-to-digital conversion.

Register 12: ADC Sample Sequence Control 0 (ADCSSCTL0), offset 0x044

This register contains the configuration information for each sample for a sequence executed with a sample sequencer. When configuring a sample sequence, the END bit must be set at some point, whether it be after the first sample, last sample, or any sample in between. This register is 32-bits wide and contains information for eight possible samples.

ADC Sample Sequence Control 0 (ADCSSCTL0)

Base 0x4003.8000
Offset 0x044
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31	TS7	R/W	0	8th Sample Temp Sensor Select This bit is used during the eighth sample of the sample sequence and and specifies the input source of the sample. When set, the temperature sensor is read. When clear, the input pin specified by the **ADCSSMUX** register is read.
30	IE7	R/W	0	8th Sample Interrupt Enable This bit is used during the eighth sample of the sample sequence and specifies whether the raw interrupt signal (INR0 bit) is asserted at the end of the sample's conversion. If the MASK0 bit in the **ADCIM** register is set, the interrupt is promoted to a controller-level interrupt. When this bit is set, the raw interrupt is asserted. When this bit is clear, the raw interrupt is not asserted. It is legal to have multiple samples within a sequence generate interrupts.
29	END7	R/W	0	8th Sample is End of Sequence The END7 bit indicates that this is the last sample of the sequence. It is possible to end the sequence on any sample position. Samples defined after the sample containing a set END are not requested for conversion even though the fields may be non-zero. It is required that software write the END bit somewhere within the sequence. (Sample Sequencer 3, which only has a single sample in the sequence, is hardwired to have the END0 bit set.) Setting this bit indicates that this sample is the last in the sequence.
28	D7	R/W	0	8th Sample Diff Input Select The D7 bit indicates that the analog input is to be differentially sampled. The corresponding **ADCSSMUXx** nibble must be set to the pair number "i", where the paired inputs are "2i and 2i+1". The temperature sensor does not have a differential option. When set, the analog inputs are differentially sampled.

Bit/Field	Name	Type	Reset	Description
27	TS6	R/W	0	7th Sample Temp Sensor Select Same definition as TS7 but used during the seventh sample.
26	IE6	R/W	0	7th Sample Interrupt Enable Same definition as IE7 but used during the seventh sample.
25	END6	R/W	0	7th Sample is End of Sequence Same definition as END7 but used during the seventh sample.
24	D6	R/W	0	7th Sample Diff Input Select Same definition as D7 but used during the seventh sample.
23	TS5	R/W	0	6th Sample Temp Sensor Select Same definition as TS7 but used during the sixth sample.
22	IE5	R/W	0	6th Sample Interrupt Enable Same definition as IE7 but used during the sixth sample.
21	END5	R/W	0	6th Sample is End of Sequence Same definition as END7 but used during the sixth sample.
20	D5	R/W	0	6th Sample Diff Input Select Same definition as D7 but used during the sixth sample.
19	TS4	R/W	0	5th Sample Temp Sensor Select Same definition as TS7 but used during the fifth sample.
18	IE4	R/W	0	5th Sample Interrupt Enable Same definition as IE7 but used during the fifth sample.
17	END4	R/W	0	5th Sample is End of Sequence Same definition as END7 but used during the fifth sample.
16	D4	R/W	0	5th Sample Diff Input Select Same definition as D7 but used during the fifth sample.
15	TS3	R/W	0	4th Sample Temp Sensor Select Same definition as TS7 but used during the fourth sample.
14	IE3	R/W	0	4th Sample Interrupt Enable Same definition as IE7 but used during the fourth sample.
13	END3	R/W	0	4th Sample is End of Sequence Same definition as END7 but used during the fourth sample.
12	D3	R/W	0	4th Sample Diff Input Select Same definition as D7 but used during the fourth sample.

Bit/Field	Name	Type	Reset	Description
11	TS2	R/W	0	3rd Sample Temp Sensor Select Same definition as TS7 but used during the third sample.
10	IE2	R/W	0	3rd Sample Interrupt Enable Same definition as IE7 but used during the third sample.
9	END2	R/W	0	3rd Sample is End of Sequence Same definition as END7 but used during the third sample.
8	D2	R/W	0	3rd Sample Diff Input Select Same definition as D7 but used during the third sample.
7	TS1	R/W	0	2nd Sample Temp Sensor Select Same definition as TS7 but used during the second sample.
6	IE1	R/W	0	2nd Sample Interrupt Enable Same definition as IE7 but used during the second sample.
5	END1	R/W	0	2nd Sample is End of Sequence Same definition as END7 but used during the second sample.
4	D1	R/W	0	2nd Sample Diff Input Select Same definition as D7 but used during the second sample.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as TS7 but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as IE7 but used during the first sample.
1	END0	R/W	0	1st Sample is End of Sequence Same definition as END7 but used during the first sample. Since this sequencer has only one entry, this bit must be set.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as D7 but used during the first sample.

Register 13: ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0), offset 0x048

Register 14: ADC Sample Sequence Result FIFO 1 (ADCSSFIFO1), offset 0x068

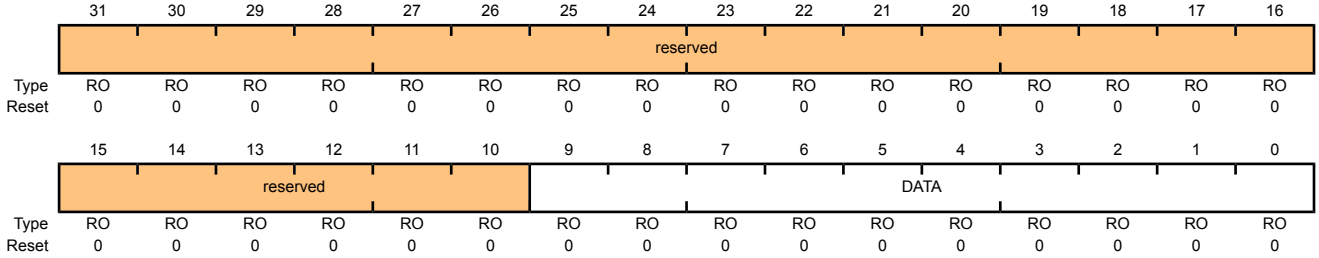
Register 15: ADC Sample Sequence Result FIFO 2 (ADCSSFIFO2), offset 0x088

Register 16: ADC Sample Sequence Result FIFO 3 (ADCSSFIFO3), offset 0x0A8

This register contains the conversion results for samples collected with the sample sequencer (the **ADCSSFIFO0** register is used for Sample Sequencer 0, **ADCSSFIFO1** for Sequencer 1, **ADCSSFIFO2** for Sequencer 2, and **ADCSSFIFO3** for Sequencer 3). Reads of this register return conversion result data in the order sample 0, sample 1, and so on, until the FIFO is empty. If the FIFO is not properly handled by software, overflow and underflow conditions are registered in the **ADCOSTAT** and **ADCUSTAT** registers.

ADC Sample Sequence Result FIFO 0 (ADCSSFIFO0)

Base 0x4003.8000
Offset 0x048
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x0000.0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:0	DATA	RO	0x000	Conversion Result Data

Register 17: ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0), offset 0x04C

Register 18: ADC Sample Sequence FIFO 1 Status (ADCSSFSTAT1), offset 0x06C

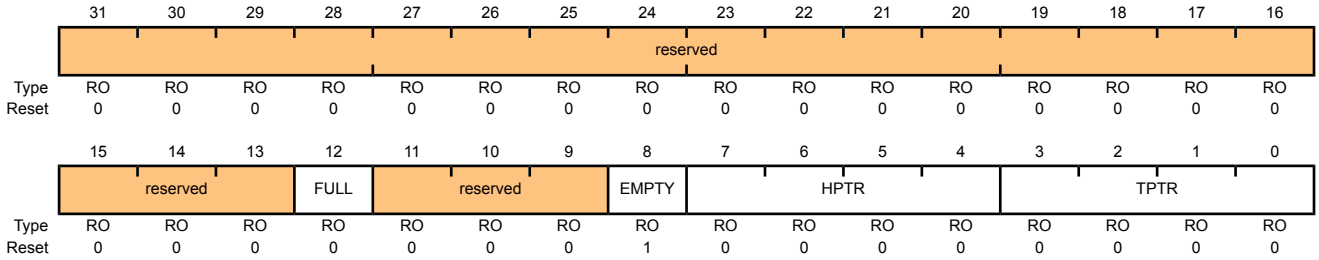
Register 19: ADC Sample Sequence FIFO 2 Status (ADCSSFSTAT2), offset 0x08C

Register 20: ADC Sample Sequence FIFO 3 Status (ADCSSFSTAT3), offset 0x0AC

This register provides a window into the sample sequencer, providing full/empty status information as well as the positions of the head and tail pointers. The reset value of 0x100 indicates an empty FIFO. The **ADCSSFSTAT0** register provides status on FIFO0, **ADCSSFSTAT1** on FIFO1, **ADCSSFSTAT2** on FIFO2, and **ADCSSFSTAT3** on FIFO3.

ADC Sample Sequence FIFO 0 Status (ADCSSFSTAT0)

Base 0x4003.8000
Offset 0x04C
Type RO, reset 0x0000.0100



Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	FULL	RO	0	FIFO Full When set, this bit indicates that the FIFO is currently full.
11:9	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
8	EMPTY	RO	1	FIFO Empty When set, this bit indicates that the FIFO is currently empty.
7:4	HPTR	RO	0x0	FIFO Head Pointer This field contains the current "head" pointer index for the FIFO, that is, the next entry to be written.
3:0	TPTR	RO	0x0	FIFO Tail Pointer This field contains the current "tail" pointer index for the FIFO, that is, the next entry to be read.

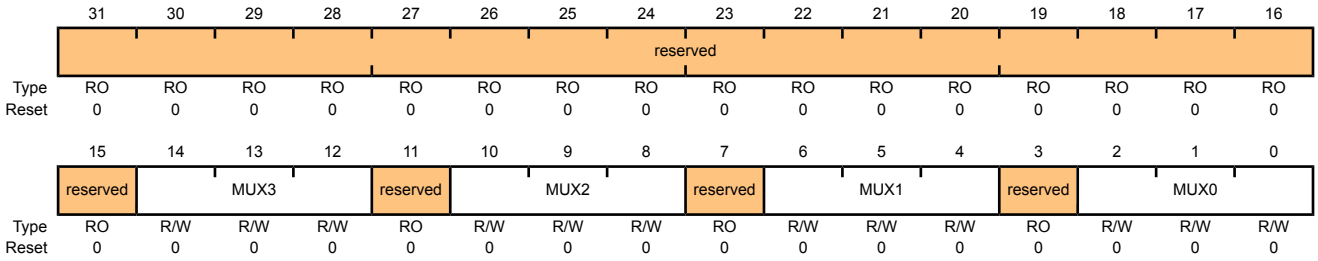
Register 21: ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1), offset 0x060

Register 22: ADC Sample Sequence Input Multiplexer Select 2 (ADCSSMUX2), offset 0x080

This register defines the analog input configuration for each sample in a sequence executed with Sample Sequencer 1 or 2. These registers are 16-bits wide and contain information for four possible samples. See the **ADCSSMUX0** register on page 384 for detailed bit descriptions. The **ADCSSMUX1** register affects Sample Sequencer 1 and the **ADCSSMUX2** register affects Sample Sequencer 2.

ADC Sample Sequence Input Multiplexer Select 1 (ADCSSMUX1)

Base 0x4003.8000
Offset 0x060
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14:12	MUX3	R/W	0x0	4th Sample Input Select
11	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10:8	MUX2	R/W	0x0	3rd Sample Input Select
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:4	MUX1	R/W	0x0	2nd Sample Input Select
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	MUX0	R/W	0x0	1st Sample Input Select

Register 23: ADC Sample Sequence Control 1 (ADCSSCTL1), offset 0x064

Register 24: ADC Sample Sequence Control 2 (ADCSSCTL2), offset 0x084

These registers contain the configuration information for each sample for a sequence executed with Sample Sequencer 1 or 2. When configuring a sample sequence, the END bit must be set at some point, whether it be after the first sample, last sample, or any sample in between. These registers are 16-bits wide and contain information for four possible samples. See the **ADCSSCTL0** register on page 386 for detailed bit descriptions. The **ADCSSCTL1** register configures Sample Sequencer 1 and the **ADCSSCTL2** register configures Sample Sequencer 2.

ADC Sample Sequence Control 1 (ADCSSCTL1)

Base 0x4003.8000
Offset 0x064
Type R/W, reset 0x0000.0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	TS3	R/W	0	4th Sample Temp Sensor Select Same definition as TS7 but used during the fourth sample.
14	IE3	R/W	0	4th Sample Interrupt Enable Same definition as IE7 but used during the fourth sample.
13	END3	R/W	0	4th Sample is End of Sequence Same definition as END7 but used during the fourth sample.
12	D3	R/W	0	4th Sample Diff Input Select Same definition as D7 but used during the fourth sample.
11	TS2	R/W	0	3rd Sample Temp Sensor Select Same definition as TS7 but used during the third sample.
10	IE2	R/W	0	3rd Sample Interrupt Enable Same definition as IE7 but used during the third sample.
9	END2	R/W	0	3rd Sample is End of Sequence Same definition as END7 but used during the third sample.
8	D2	R/W	0	3rd Sample Diff Input Select Same definition as D7 but used during the third sample.

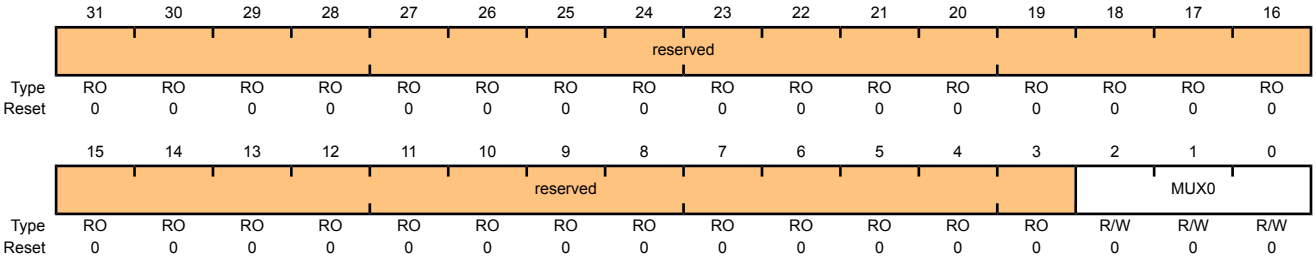
Bit/Field	Name	Type	Reset	Description
7	TS1	R/W	0	2nd Sample Temp Sensor Select Same definition as TS7 but used during the second sample.
6	IE1	R/W	0	2nd Sample Interrupt Enable Same definition as IE7 but used during the second sample.
5	END1	R/W	0	2nd Sample is End of Sequence Same definition as END7 but used during the second sample.
4	D1	R/W	0	2nd Sample Diff Input Select Same definition as D7 but used during the second sample.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as TS7 but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as IE7 but used during the first sample.
1	END0	R/W	0	1st Sample is End of Sequence Same definition as END7 but used during the first sample. Since this sequencer has only one entry, this bit must be set.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as D7 but used during the first sample.

Register 25: ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3), offset 0x0A0

This register defines the analog input configuration for a sample executed with Sample Sequencer 3. This register is 4-bits wide and contains information for one possible sample. See the **ADCSSMUX0** register on page 384 for detailed bit descriptions.

ADC Sample Sequence Input Multiplexer Select 3 (ADCSSMUX3)

Base 0x4003.8000
Offset 0x0A0
Type R/W, reset 0x0000.0000



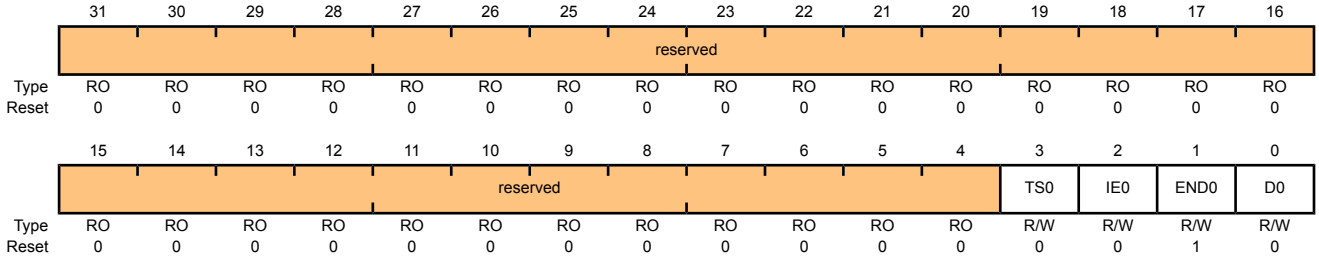
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2:0	MUX0	R/W	0	1st Sample Input Select

Register 26: ADC Sample Sequence Control 3 (ADCSSCTL3), offset 0x0A4

This register contains the configuration information for a sample executed with Sample Sequencer 3. The END bit is always set since there is only one sample in this sequencer. This register is 4-bits wide and contains information for one possible sample. See the **ADCSSCTL0** register on page 386 for detailed bit descriptions.

ADC Sample Sequence Control 3 (ADCSSCTL3)

Base 0x4003.8000
Offset 0x0A4
Type R/W, reset 0x0000.0002



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TS0	R/W	0	1st Sample Temp Sensor Select Same definition as TS7 but used during the first sample.
2	IE0	R/W	0	1st Sample Interrupt Enable Same definition as IE7 but used during the first sample.
1	END0	R/W	1	1st Sample is End of Sequence Same definition as END7 but used during the first sample. Since this sequencer has only one entry, this bit must be set.
0	D0	R/W	0	1st Sample Diff Input Select Same definition as D7 but used during the first sample.

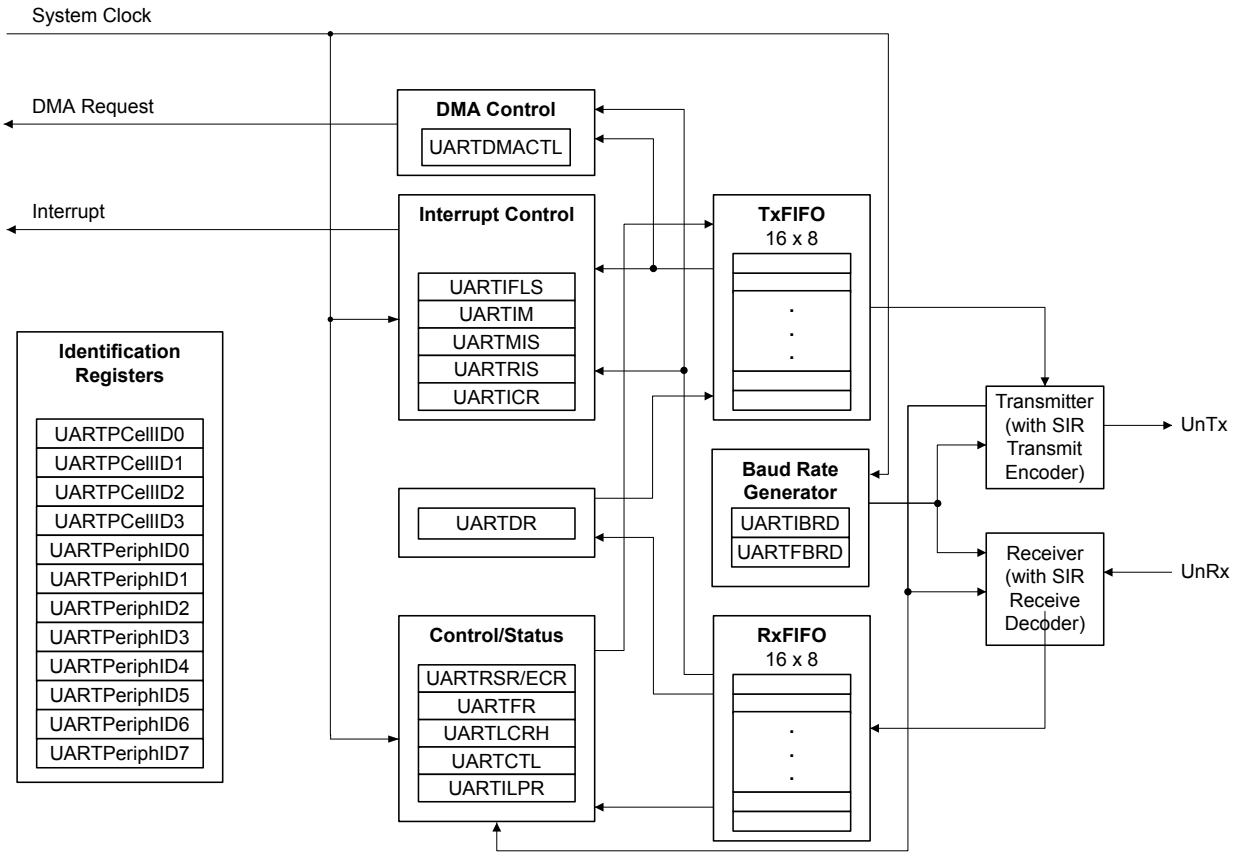
14 Universal Asynchronous Receivers/Transmitters (UARTs)

Each Stellaris® Universal Asynchronous Receiver/Transmitter (UART) has the following features:

- Two fully programmable 16C550-type UARTs with IrDA support
- Separate 16x8 transmit (TX) and 16x12 receive (RX) FIFOs to reduce CPU interrupt service loading
- Programmable baud-rate generator allowing speeds up to 3.125 Mbps
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
- Standard asynchronous communication bits for start, stop, and parity
- False-start bit detection
- Line-break generation and detection
- Fully programmable serial interface characteristics
  - 5, 6, 7, or 8 data bits
  - Even, odd, stick, or no-parity bit generation/detection
  - 1 or 2 stop bit generation
- IrDA serial-IR (SIR) encoder/decoder providing
  - Programmable use of IrDA Serial Infrared (SIR) or UART input/output
  - Support of IrDA SIR encoder/decoder functions for data rates up to 115.2 Kbps half-duplex
  - Support of normal 3/16 and low-power (1.41-2.23 µs) bit durations
  - Programmable internal clock generator enabling division of reference clock by 1 to 256 for low-power mode bit duration
- Dedicated Direct Memory Access (DMA) transmit and receive channels

14.1 Block Diagram

Figure 14-1. UART Module Block Diagram



14.2 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 416). Transmit and receive are both enabled out of reset. Before any control registers are programmed, the UART must be disabled by clearing the UARTEN bit in **UARTCTL**. If the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

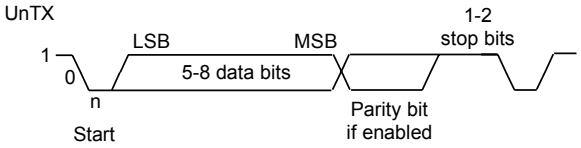
The UART peripheral also includes a serial IR (SIR) encoder/decoder block that can be connected to an infrared transceiver to implement an IrDA SIR physical layer. The SIR function is programmed using the UARTCTL register.

14.2.1 Transmit/Receive Logic

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. The control logic outputs the serial bit stream beginning with a start bit, and followed by the data bits (LSB first), parity bit, and the stop bits according to the programmed configuration in the control registers. See Figure 14-2 on page 398 for details.

The receive logic performs serial-to-parallel conversion on the received bit stream after a valid start pulse has been detected. Overrun, parity, frame error checking, and line-break detection are also performed, and their status accompanies the data that is written to the receive FIFO.

Figure 14-2. UART Character Frame



14.2.2 Baud-Rate Generation

The baud-rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. The number formed by these two values is used by the baud-rate generator to determine the bit period. Having a fractional baud-rate divider allows the UART to generate all the standard baud rates.

The 16-bit integer is loaded through the **UART Integer Baud-Rate Divisor (UARTIBRD)** register (see page 412) and the 6-bit fractional part is loaded with the **UART Fractional Baud-Rate Divisor (UARTFBRD)** register (see page 413). The baud-rate divisor (BRD) has the following relationship to the system clock (where *BRDI* is the integer part of the BRD and *BRDF* is the fractional part, separated by a decimal place.)

```
BRD = BRDI + BRDF = UARTSysClk / (16 * Baud Rate)
```

where UARTSysClk is the system clock connected to the UART.

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 414), the **UARTIBRD** and **UARTFBRD** registers form an internal 30-bit register. This internal register is only updated when a write operation to **UARTLCRH** is performed, so any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register for the changes to take effect.

To update the baud-rate registers, there are four possible sequences:

- **UARTIBRD** write, **UARTFBRD** write, and **UARTLCRH** write
- **UARTFBRD** write, **UARTIBRD** write, and **UARTLCRH** write
- **UARTIBRD** write and **UARTLCRH** write
- **UARTFBRD** write and **UARTLCRH** write

14.2.3 Data Transmission

Data received or transmitted is stored in two 16-byte FIFOs, though the receive FIFO has an extra four bits per character for status information. For transmission, data is written into the transmit FIFO. If the UART is enabled, it causes a data frame to start transmitting with the parameters indicated in the **UARTLCRH** register. Data continues to be transmitted until there is no data left in the transmit FIFO. The BUSY bit in the **UART Flag (UARTFR)** register (see page 409) 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 397).

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 407). 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.

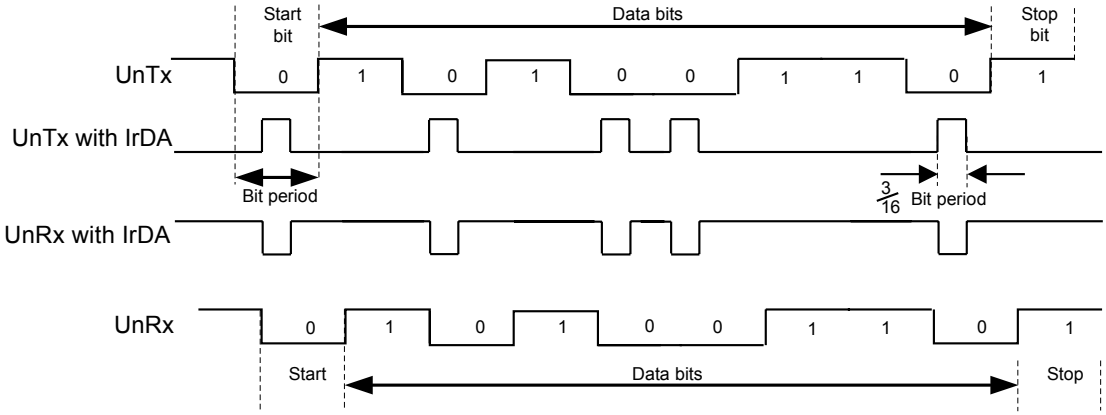
14.2.4 Serial IR (SIR)

The UART peripheral includes an IrDA serial-IR (SIR) encoder/decoder block. The IrDA SIR block provides functionality that converts between an asynchronous UART data stream, and half-duplex serial SIR interface. No analog processing is performed on-chip. The role of the SIR block is to provide a digital encoded output, and decoded input to the UART. The UART signal pins can be connected to an infrared transceiver to implement an IrDA SIR physical layer link. The SIR block has two modes of operation:

- In normal IrDA mode, a zero logic level is transmitted as high pulse of 3/16th duration of the selected baud rate bit period on the output pin, while logic one levels are transmitted as a static LOW signal. These levels control the driver of an infrared transmitter, sending a pulse of light for each zero. On the reception side, the incoming light pulses energize the photo transistor base of the receiver, pulling its output LOW. This drives the UART input pin LOW.
- In low-power IrDA mode, the width of the transmitted infrared pulse is set to three times the period of the internally generated IrLPBaud16 signal (1.63 µs, assuming a nominal 1.8432 MHz frequency) by changing the appropriate bit in the **UARTCR** register. See page 411 for more information on IrDA low-power pulse-duration configuration.

Figure 14-3 on page 400 shows the UART transmit and receive signals, with and without IrDA modulation.

Figure 14-3. IrDA Data Modulation



In both normal and low-power IrDA modes:

- During transmission, the UART data bit is used as the base for encoding
- During reception, the decoded bits are transferred to the UART receive logic

The IrDA SIR physical layer specifies a half-duplex communication link, with a minimum 10 ms delay between transmission and reception. This delay must be generated by software because it is not automatically supported by the UART. The delay is required because the infrared receiver electronics might become biased, or even saturated from the optical power coupled from the adjacent transmitter LED. This delay is known as latency, or receiver setup time.

14.2.5 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 405). 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 414).

FIFO status can be monitored via the **UART Flag (UARTFR)** register (see page 409) 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 418). 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.

14.2.6 Interrupts

The UART can generate interrupts when the following conditions are observed:

- Overrun Error

- Break Error
- Parity Error
- Framing Error
- Receive Timeout
- Transmit (when condition defined in the TXIFLSEL bit in the **UARTIFLS** register is met)
- Receive (when condition defined in the RXIFLSEL bit in the **UARTIFLS** register is met)

All of the interrupt events are ORed together before being sent to the interrupt controller, so the UART can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine by reading the **UART Masked Interrupt Status (UARTMIS)** register (see page 423).

The interrupt events that can trigger a controller-level interrupt are defined in the **UART Interrupt Mask (UARTIM**) register (see page 420) 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 422).

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 424).

The receive timeout interrupt is asserted when the receive FIFO is not empty, and no further data is received over a 32-bit period. The receive timeout interrupt is cleared either when the FIFO becomes empty through reading all the data (or by reading the holding register), or when a 1 is written to the corresponding bit in the **UARTICR** register.

14.2.7 Loopback Operation

The UART can be placed into an internal loopback mode for diagnostic or debug work. This is accomplished by setting the LBE bit in the **UARTCTL** register (see page 416). In loopback mode, data transmitted on UnTx is received on the UnRx input.

14.2.8 DMA Operation

The UART provides an interface connected to the µDMA controller. The DMA operation of the UART is enabled through the **UART DMA Control (UARTDMACTL)** register. When DMA operation is enabled, the UART will assert a DMA request on the receive or transmit channel when the associated FIFO can transfer data. For the receive channel, a single transfer request is asserted whenever there is any data in the receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is at or above the FIFO trigger level. For the transmit channel, a single transfer request is asserted whenever there is at least one empty location in the transmit FIFO. The burst request is asserted whenever the transmit FIFO contains fewer characters than the FIFO trigger level. The single and burst DMA transfer requests are handled automatically by the µDMA controller depending how the DMA channel is configured.

To enable DMA operation for the receive channel, the RXDMAE bit of the **DMA Control (UARTDMACTL)** register should be set. To enable DMA operation for the transmit channel, the TXDMAE bit of **UARTDMACTL** should be set. The UART can also be configured to stop using DMA for the receive channel if a receive error occurs. If the DMAERR bit of **UARTDMACR** is set, then when a receive error occurs, the DMA receive requests will be automatically disabled. This error condition can be cleared by clearing the UART error interrupt.

If DMA is enabled, then the µDMA controller will trigger an interrupt when a transfer is complete. The interrupt will occur on the UART interrupt vector. Therefore, if interrupts are used for UART operation and DMA is enabled, the UART interrupt handler must be designed to handle the µDMA completion interrupt.

See “Micro Direct Memory Access (µDMA)” on page 194 for more details about programming the µDMA controller.

14.2.9 IrDA SIR block

The IrDA SIR block contains an IrDA serial IR (SIR) protocol encoder/decoder. When enabled, the SIR block uses the UnTx and UnRx pins for the SIR protocol, which should be connected to an IR transceiver.

The SIR block can receive and transmit, but it is only half-duplex so it cannot do both at the same time. Transmission must be stopped before data can be received. The IrDA SIR physical layer specifies a minimum 10-ms delay between transmission and reception.

14.3 Initialization and Configuration

To use the UARTs, the peripheral clock must be enabled by setting the UART0 or UART1 bits in the **RCGC1** register.

This section discusses the steps that are required to use a UART module. For this example, the UART clock is assumed to be 20 MHz and the desired UART configuration is:

- 115200 baud rate
- Data length of 8 bits
- One stop bit
- No parity
- FIFOs disabled
- No interrupts

The first thing to consider when programming the UART is the baud-rate divisor (BRD), since the **UARTIBRD** and **UARTFBRD** registers must be written before the **UARTLCRH** register. Using the equation described in “Baud-Rate Generation” on page 398, 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 412) should be set to 10. The value to be loaded into the **UARTFBRD** register (see page 413) is calculated by the equation:

```
UARTFBRD[DIVFRAC] = integer(0.8507 * 64 + 0.5) = 54
```

With the BRD values in hand, the UART configuration is written to the module in the following order:

1. Disable the UART by clearing the UARTEN bit in the **UARTCTL** register.
2. Write the integer portion of the BRD to the **UARTIBRD** register.
3. Write the fractional portion of the BRD to the **UARTFBRD** register.

4. Write the desired serial parameters to the **UARTLCRH** register (in this case, a value of 0x0000.0060).

5. Optionally, configure the uDMA channel (see "Micro Direct Memory Access (µDMA)" on page 194) and enable the DMA option(s) in the **UARTDMACTL** register.

6. Enable the UART by setting the UARTEN bit in the **UARTCTL** register.

14.4 Register Map

Table 14-1 on page 403 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 416) before any of the control registers are reprogrammed. When the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

Table 14-1. UART Register Map

Offset	Name	Type	Reset	Description	See page
0x000	UARTDR	R/W	0x0000.0000	UART Data	405
0x004	UARTRSR/UARTECR	R/W	0x0000.0000	UART Receive Status/Error Clear	407
0x018	UARTFR	RO	0x0000.0090	UART Flag	409
0x020	UARTILPR	R/W	0x0000.0000	UART IrDA Low-Power Register	411
0x024	UARTIBRD	R/W	0x0000.0000	UART Integer Baud-Rate Divisor	412
0x028	UARTFBRD	R/W	0x0000.0000	UART Fractional Baud-Rate Divisor	413
0x02C	UARTLCRH	R/W	0x0000.0000	UART Line Control	414
0x030	UARTCTL	R/W	0x0000.0300	UART Control	416
0x034	UARTIFLS	R/W	0x0000.0012	UART Interrupt FIFO Level Select	418
0x038	UARTIM	R/W	0x0000.0000	UART Interrupt Mask	420
0x03C	UARTRIS	RO	0x0000.000F	UART Raw Interrupt Status	422
0x040	UARTMIS	RO	0x0000.0000	UART Masked Interrupt Status	423
0x044	UARTICR	W1C	0x0000.0000	UART Interrupt Clear	424
0x048	UARTDMACTL	R/W	0x0000.0000	UART DMA Control	426
0xFD0	UARTPeriphID4	RO	0x0000.0000	UART Peripheral Identification 4	427
0xFD4	UARTPeriphID5	RO	0x0000.0000	UART Peripheral Identification 5	428
0xFD8	UARTPeriphID6	RO	0x0000.0000	UART Peripheral Identification 6	429
0xFDC	UARTPeriphID7	RO	0x0000.0000	UART Peripheral Identification 7	430
0xFE0	UARTPeriphID0	RO	0x0000.0011	UART Peripheral Identification 0	431

Offset	Name	Type	Reset	Description	See page
0xFE4	UARTPeriphID1	RO	0x0000.0000	UART Peripheral Identification 1	432
0xFE8	UARTPeriphID2	RO	0x0000.0018	UART Peripheral Identification 2	433
0xFEC	UARTPeriphID3	RO	0x0000.0001	UART Peripheral Identification 3	434
0xFF0	UARTPCellID0	RO	0x0000.000D	UART PrimeCell Identification 0	435
0xFF4	UARTPCellID1	RO	0x0000.00F0	UART PrimeCell Identification 1	436
0xFF8	UARTPCellID2	RO	0x0000.0005	UART PrimeCell Identification 2	437
0xFFC	UARTPCellID3	RO	0x0000.00B1	UART PrimeCell Identification 3	438

14.5 Register Descriptions

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

Register 1: UART Data (UARTDR), offset 0x000

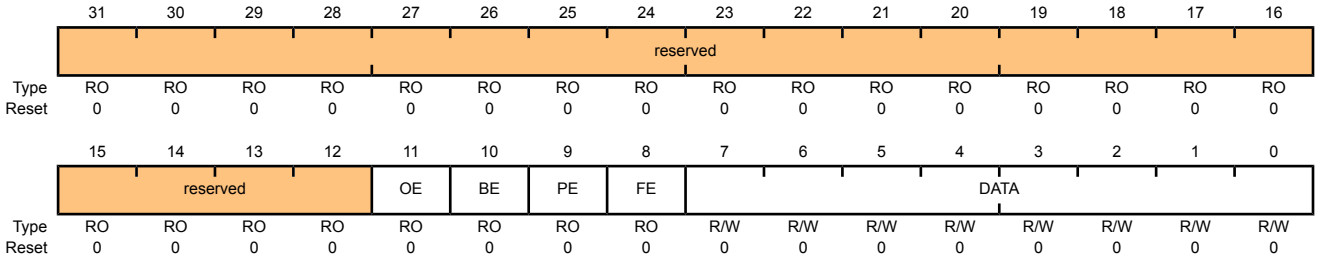
This register is the data register (the interface to the FIFOs).

When FIFOs are enabled, data written to this location is pushed onto the transmit FIFO. If FIFOs are disabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). A write to this register initiates a transmission from the UART.

For received data, if the FIFO is enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO. If FIFOs are disabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO). The received data can be retrieved by reading this register.

UART Data (UARTDR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x000
Type R/W, reset 0x0000.0000



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.

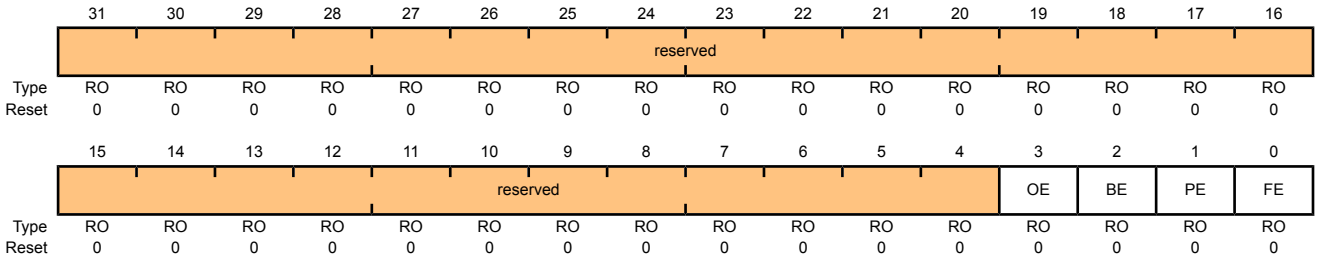
The **UARTRSR** register cannot be written.

A write of any value to the **UARTECR** register clears the framing, parity, break, and overrun errors. All the bits are cleared to 0 on reset.

Reads

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x004
Type RO, reset 0x0000.0000



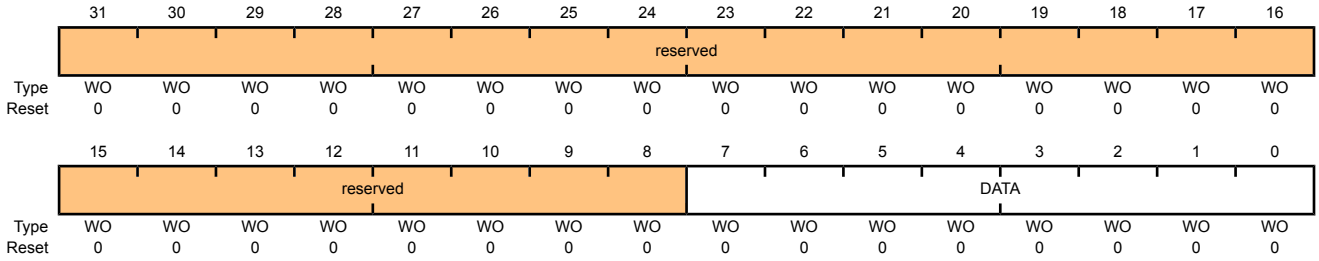
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	OE	RO	0	UART Overrun Error When this bit is set to 1, 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.

Writes

UART Receive Status/Error Clear (UARTRSR/UARTECR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x004
Type WO, reset 0x0000.0000

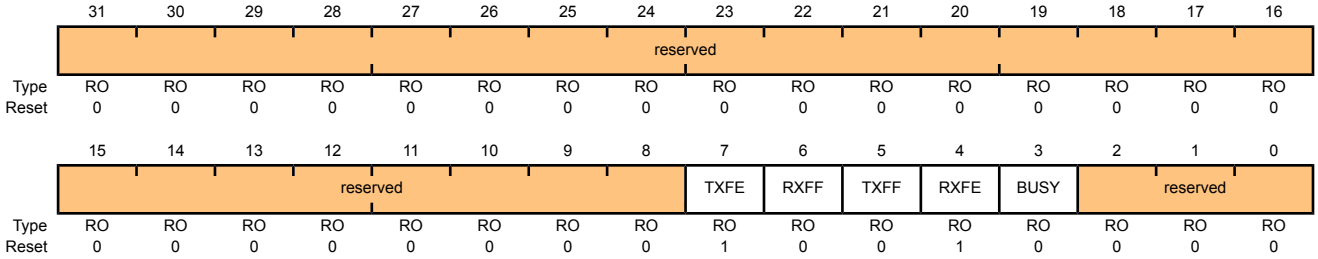


Bit/Field	Name	Type	Reset	Description
31:8	reserved	WO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	WO	0	Error Clear A write to this register of 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



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 IrDA Low-Power Register (UARTILPR), offset 0x020

The **UARTILPR** register is an 8-bit read/write register that stores the low-power counter divisor value used to derive the low-power SIR pulse width clock by dividing down the system clock (SysClk). All the bits are cleared to 0 when reset.

The internal `IrLPBaud16` clock is generated by dividing down SysClk according to the low-power divisor value written to **UARTILPR**. The duration of SIR pulses generated when low-power mode is enabled is three times the period of the `IrLPBaud16` clock. The low-power divisor value is calculated as follows:

$$ILPDVSR = SysClk / F_{IrLPBaud16}$$

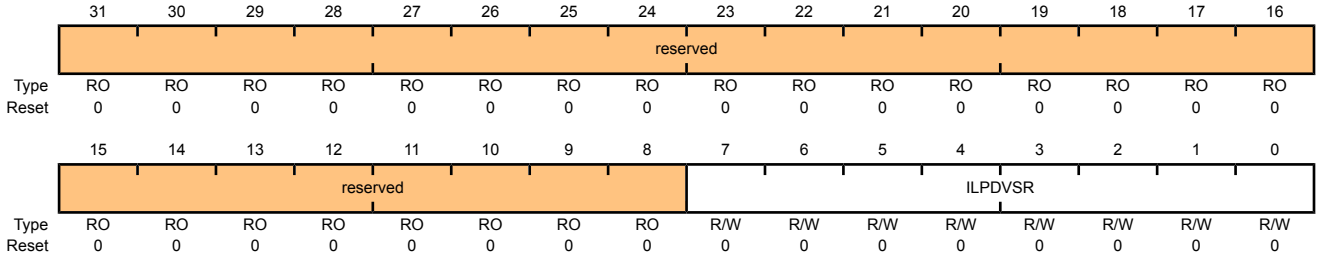
where $F_{IrLPBaud16}$ is nominally 1.8432 MHz.

You must choose the divisor so that 1.42 MHz < $F_{IrLPBaud16}$ < 2.12 MHz, which results in a low-power pulse duration of 1.41–2.11 µs (three times the period of `IrLPBaud16`). The minimum frequency of `IrLPBaud16` ensures that pulses less than one period of `IrLPBaud16` are rejected, but that pulses greater than 1.4 µs are accepted as valid pulses.

Note: Zero is an illegal value. Programming a zero value results in no `IrLPBaud16` pulses being generated.

UART IrDA Low-Power Register (UARTILPR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x020
Type R/W, reset 0x0000.0000



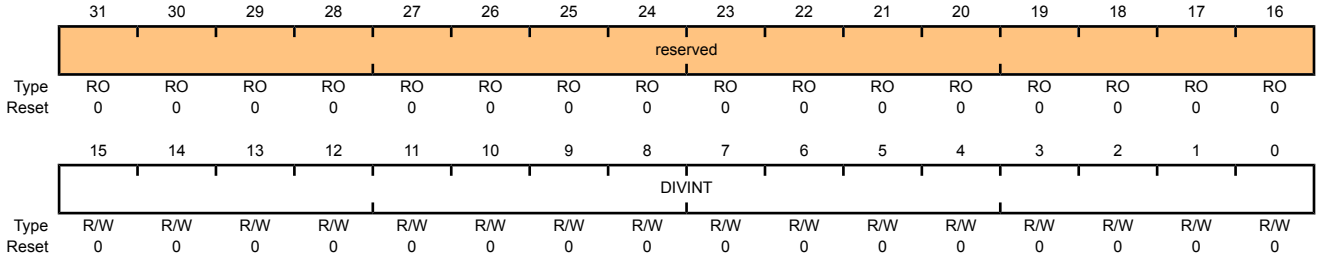
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	ILPDVSR	R/W	0x00	IrDA Low-Power Divisor This is an 8-bit low-power divisor value.

Register 5: UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024

The **UARTIBRD** register is the integer part of the baud-rate divisor value. All the bits are cleared on reset. The minimum possible divide ratio is 1 (when **UARTIBRD**=0), in which case the **UARTFBRD** register is ignored. When changing the **UARTIBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 398 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



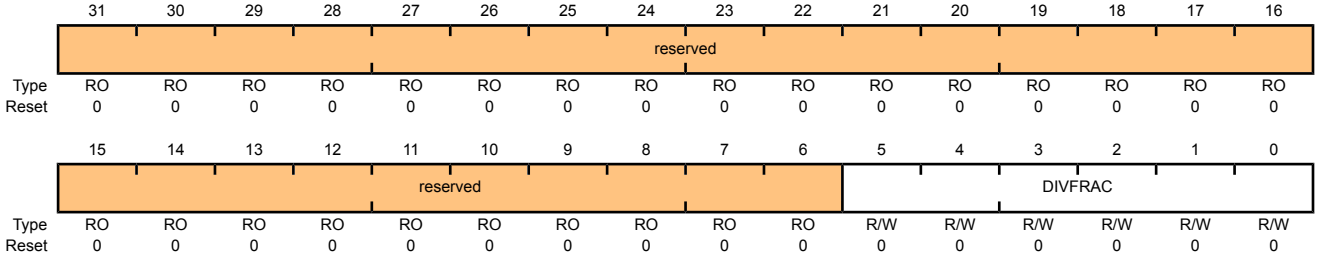
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DIVINT	R/W	0x0000	Integer Baud-Rate Divisor

Register 6: UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028

The **UARTFBRD** register is the fractional part of the baud-rate divisor value. All the bits are cleared on reset. When changing the **UARTFBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 398 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



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	DIVFRAC	R/W	0x000	Fractional Baud-Rate Divisor

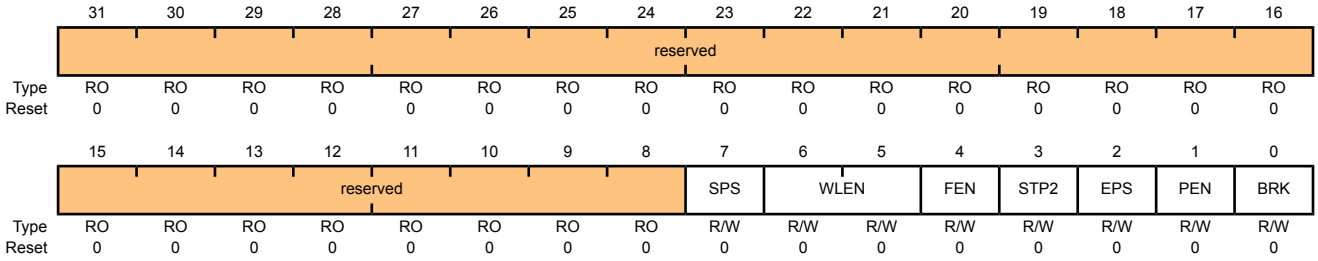
Register 7: UART Line Control (UARTLCRH), offset 0x02C

The **UARTLCRH** register is the line control register. Serial parameters such as data length, parity, and stop bit selection are implemented in this register.

When updating the baud-rate divisor (**UARTIBRD** and/or **UARTIFRD**), the **UARTLCRH** register must also be written. The write strobe for the baud-rate divisor registers is tied to the **UARTLCRH** register.

UART Line Control (UARTLCRH)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x02C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	SPS	R/W	0	UART Stick Parity Select When bits 1, 2, and 7 of **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 8: UART Control (UARTCTL), offset 0x030

The **UARTCTL** register is the control register. All the bits are cleared on reset except for the Transmit Enable (TXE) and Receive Enable (RXE) bits, which are set to 1.

To enable the UART module, the UARTEN bit must be set to 1. If software requires a configuration change in the module, the UARTEN bit must be cleared before the configuration changes are written. If the UART is disabled during a transmit or receive operation, the current transaction is completed prior to the UART stopping.

Note: The **UARTCTL** register should not be changed while the UART is enabled or else the results are unpredictable. The following sequence is recommended for making changes to the **UARTCTL** register.

1. Disable the UART.
2. Wait for the end of transmission or reception of the current character.
3. Flush the transmit FIFO by disabling bit 4 (FEN) in the line control register (**UARTLCRH**).
4. Reprogram the control register.
5. Enable the UART.

UART Control (UARTCTL)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
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				SIRLP	SIREN	UARTEN
Type RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	RO	RO	RO	RO	R/W	R/W	R/W
Reset 0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0	Software should not rely on the value of 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.

Bit/Field	Name	Type	Reset	Description
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.
6:3	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	SIRLP	R/W	0	UART SIR Low Power Mode This bit selects the IrDA encoding mode. If this bit is cleared to 0, low-level bits are transmitted as an active High pulse with a width of 3/16th of the bit period. If this bit is set to 1, low-level bits are transmitted with a pulse width which is 3 times the period of the IrLPBaud16 input signal, regardless of the selected bit rate. Setting this bit uses less power, but might reduce transmission distances. See page 411 for more information.
1	SIREN	R/W	0	UART SIR Enable If this bit is set to 1, the IrDA SIR block is enabled, and the UART will transmit and receive data using SIR protocol.
0	UARTEN	R/W	0	UART Enable If this bit is set to 1, the UART is enabled. When the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.

Register 9: UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034

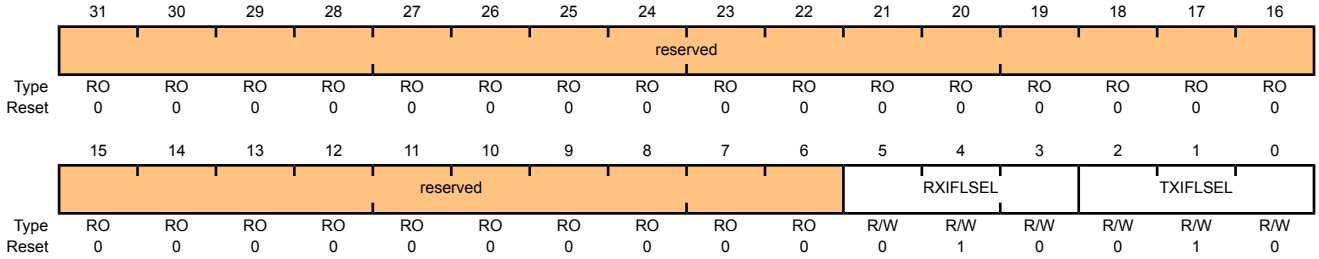
The **UARTIFLS** register is the interrupt FIFO level select register. You can use this register to define the FIFO level at which the TXRIS and RXRIS bits in the **UARTRIS** register are triggered.

The interrupts are generated based on a transition through a level rather than being based on the level. That is, the interrupts are generated when the fill level progresses through the trigger level. For example, if the receive trigger level is set to the half-way mark, the interrupt is triggered as the module is receiving the 9th character.

Out of reset, the TXIFLSEL and RXIFLSEL bits are configured so that the FIFOs trigger an interrupt at the half-way mark.

UART Interrupt FIFO Level Select (UARTIFLS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x034
Type R/W, reset 0x0000.0012



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:3	RXIFLSEL	R/W	0x2	UART Receive Interrupt FIFO Level Select The trigger points for the receive interrupt are as follows: Value Description 0x0 RX FIFO ≥ 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

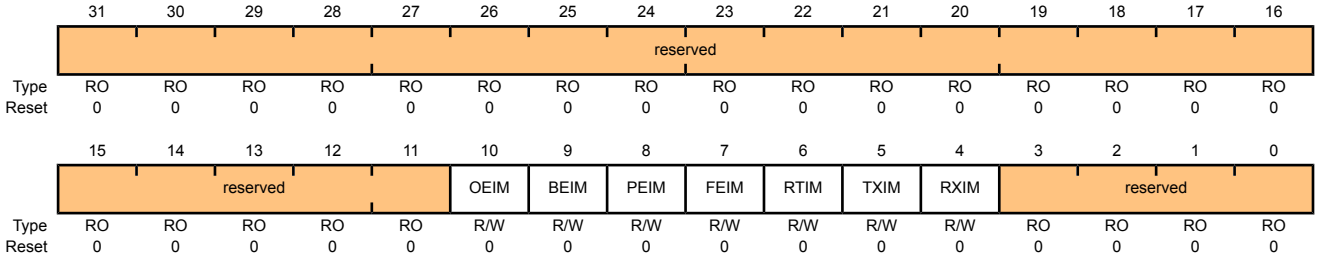
Register 10: UART Interrupt Mask (UARTIM), offset 0x038

The **UARTIM** register is the interrupt mask set/clear register.

On a read, this register gives the current value of the mask on the relevant interrupt. Writing a 1 to a bit allows the corresponding raw interrupt signal to be routed to the interrupt controller. Writing a 0 prevents the raw interrupt signal from being sent to the interrupt controller.

UART Interrupt Mask (UARTIM)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x038
Type R/W, reset 0x0000.0000



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.

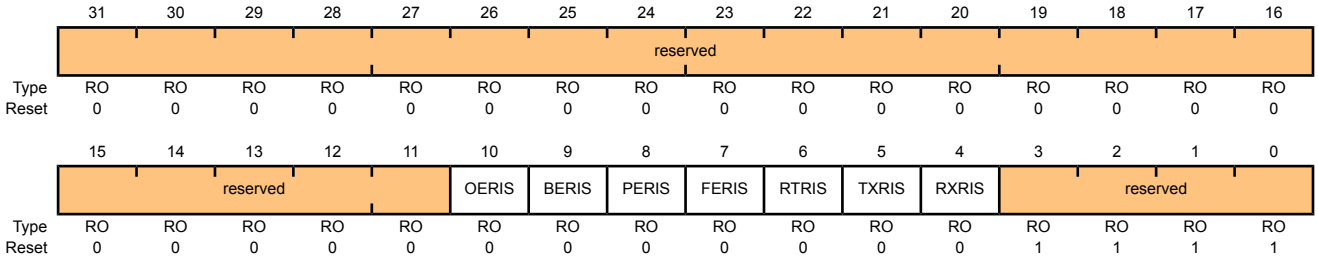
Bit/Field	Name	Type	Reset	Description
4	RXIM	R/W	0	UART Receive Interrupt Mask On a read, the current mask for the RXIM interrupt is returned. Setting this bit to 1 promotes the RXIM interrupt to the interrupt controller.
3:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 11: UART Raw Interrupt Status (UARTRIS), offset 0x03C

The **UARTRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt. A write has no effect.

UART Raw Interrupt Status (UARTRIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x03C
Type RO, reset 0x0000.000F



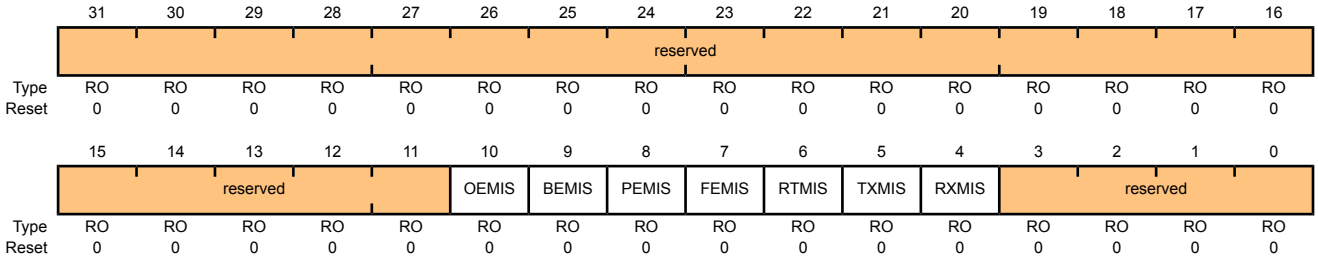
Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OERIS	RO	0	UART Overrun Error Raw Interrupt Status 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 12: UART Masked Interrupt Status (UARTMIS), offset 0x040

The **UARTMIS** register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

UART Masked Interrupt Status (UARTMIS)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x040
Type RO, reset 0x0000.0000



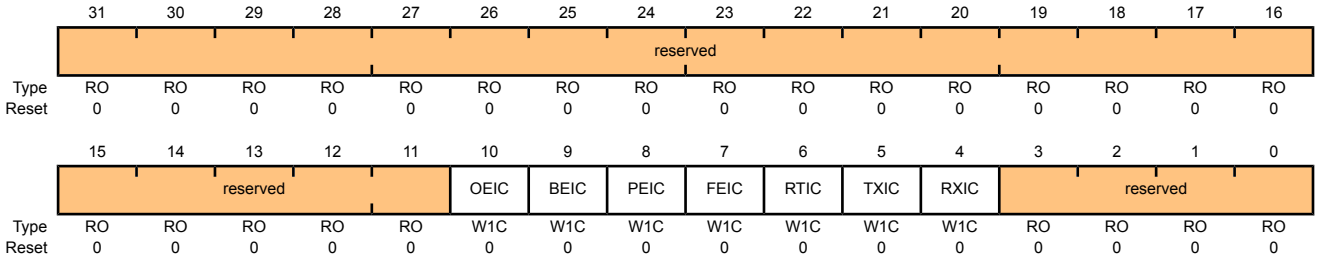
Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10	OEMIS	RO	0	UART Overrun Error Masked Interrupt 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 13: UART Interrupt Clear (UARTICR), offset 0x044

The **UARTICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt (both raw interrupt and masked interrupt, if enabled) is cleared. A write of 0 has no effect.

UART Interrupt Clear (UARTICR)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x044
Type W1C, reset 0x0000.0000



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.

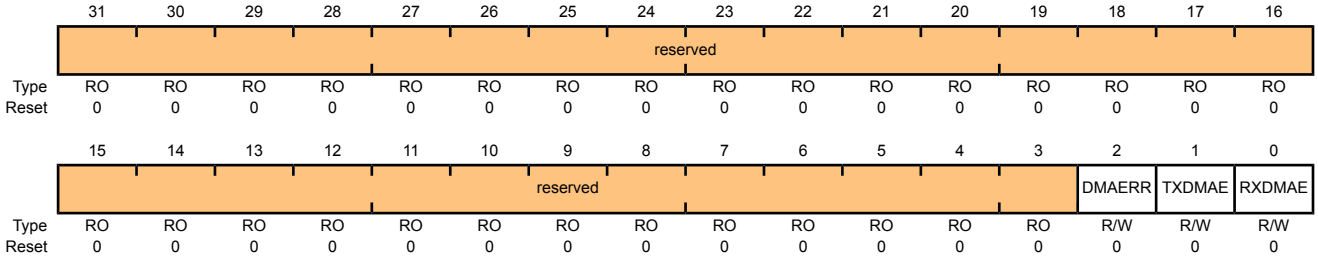
Bit/Field	Name	Type	Reset	Description
7	FEIC	W1C	0	Framing Error Interrupt Clear The FEIC values are defined as follows: Value Description 0 No effect on the interrupt. 1 Clears interrupt.
6	RTIC	W1C	0	Receive Time-Out Interrupt Clear The RTIC values are defined as follows: Value Description 0 No effect 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 14: UART DMA Control (UARTDMACTL), offset 0x048

The **UARTDMACTL** register is the DMA control register.

UART DMA Control (UARTDMACTL)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
Offset 0x048
Type R/W, reset 0x0000.0000



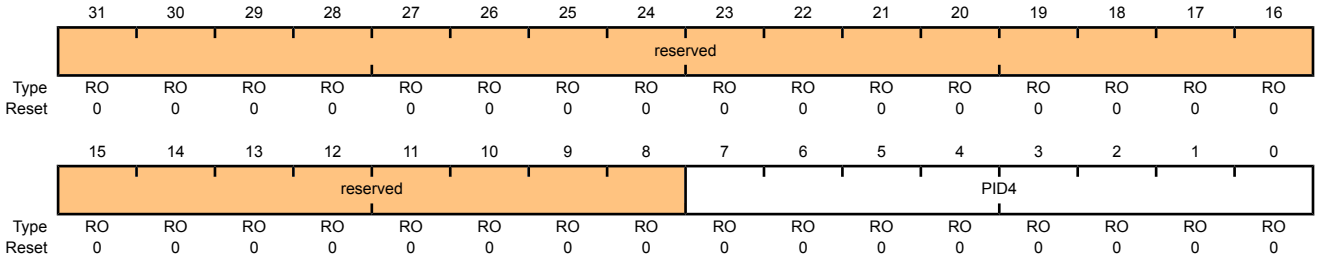
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	DMAERR	R/W	0	DMA on Error If this bit is set to 1, DMA receive requests are automatically disabled when a receive error occurs.
1	TXDMAE	R/W	0	Transmit DMA Enable If this bit is set to 1, DMA for the transmit FIFO is enabled.
0	RXDMAE	R/W	0	Receive DMA Enable If this bit is set to 1, DMA for the receive FIFO is enabled.

Register 15: UART 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



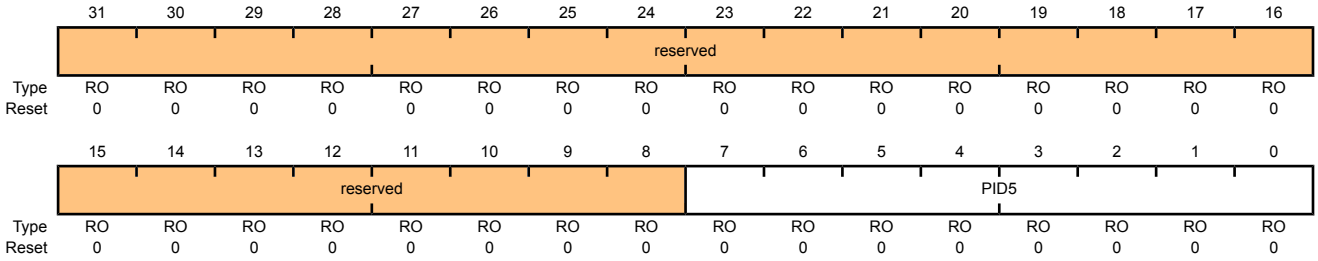
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x0000	UART Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 16: 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



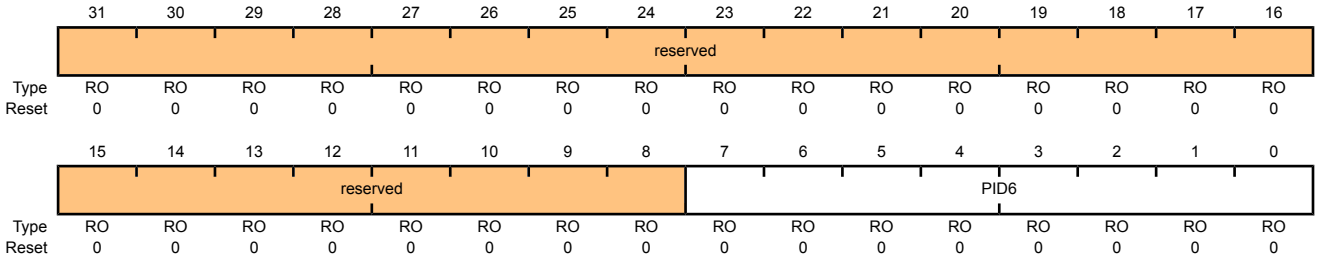
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x0000	UART Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

Register 17: 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



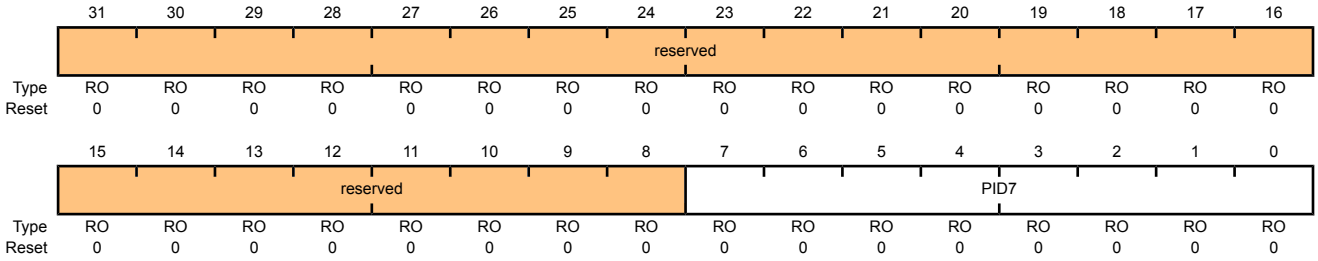
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x0000	UART Peripheral ID Register[23:16] Can be used by software to identify the presence of this peripheral.

Register 18: 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



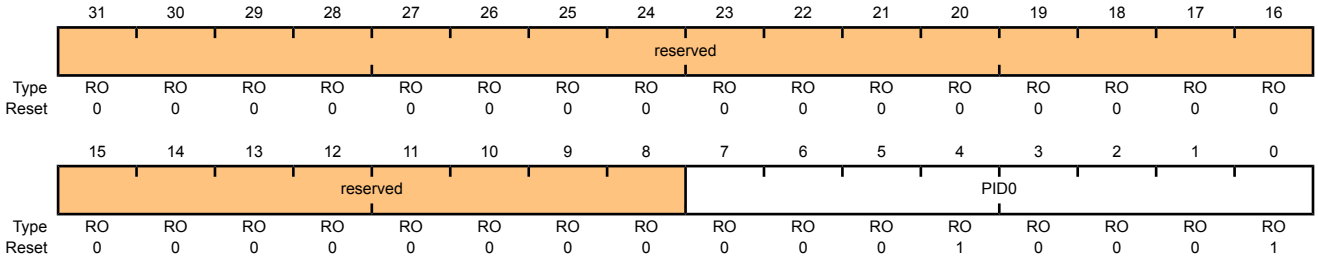
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x0000	UART Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

Register 19: 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



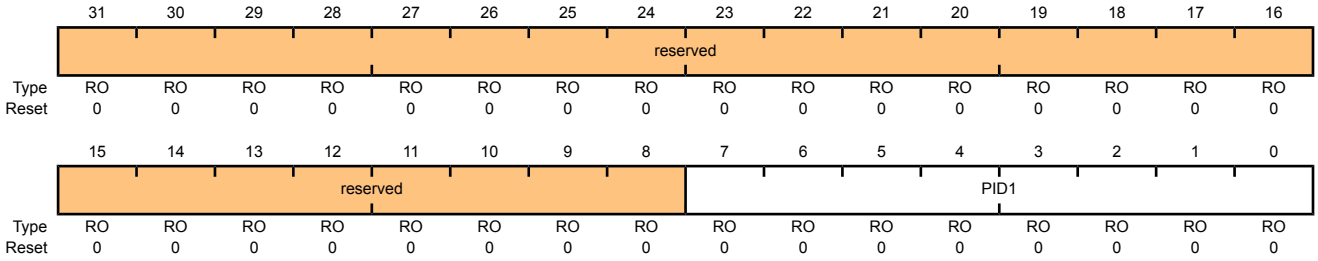
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x11	UART Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 20: 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



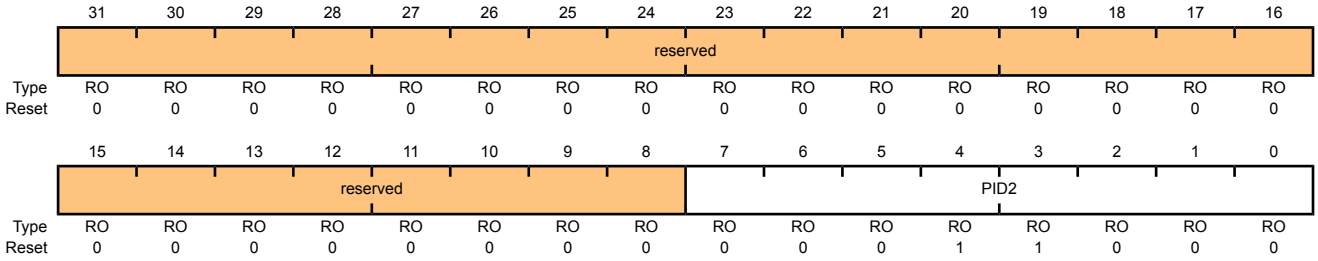
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	UART Peripheral ID Register[15:8]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

Register 21: 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



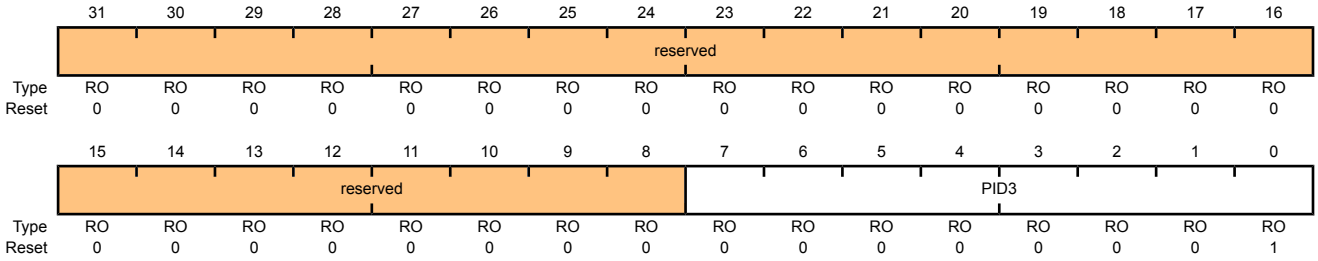
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	UART Peripheral ID Register[23:16] Can be used by software to identify the presence of this peripheral.

Register 22: 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



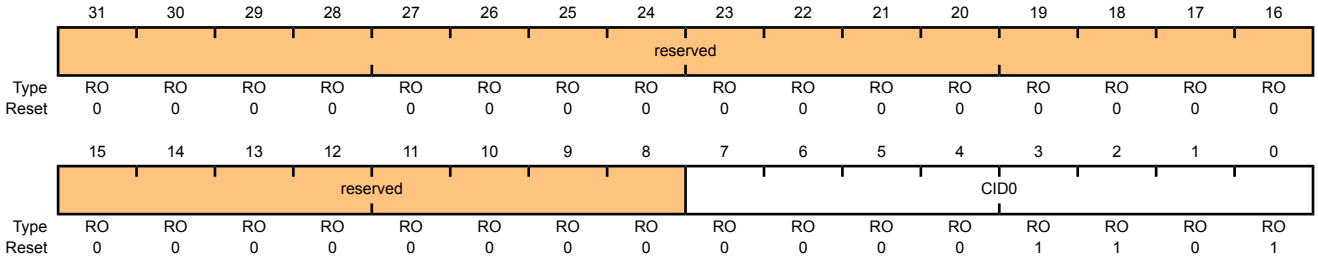
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	UART Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

Register 23: 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



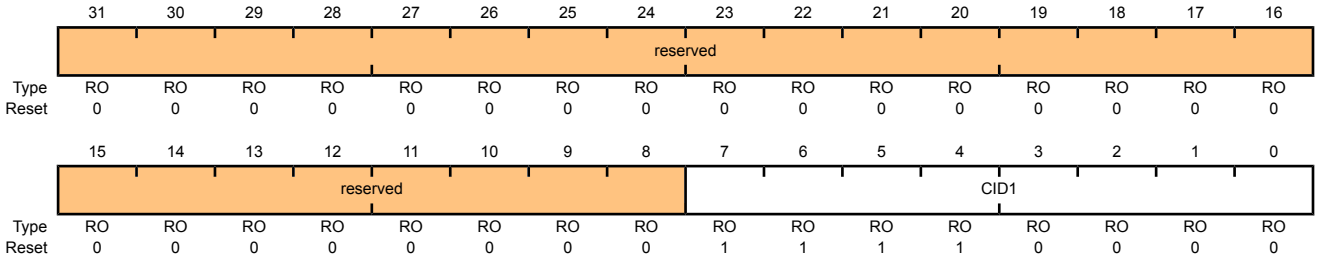
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	UART PrimeCell ID Register[7:0] Provides software a standard cross-peripheral identification system.

Register 24: 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



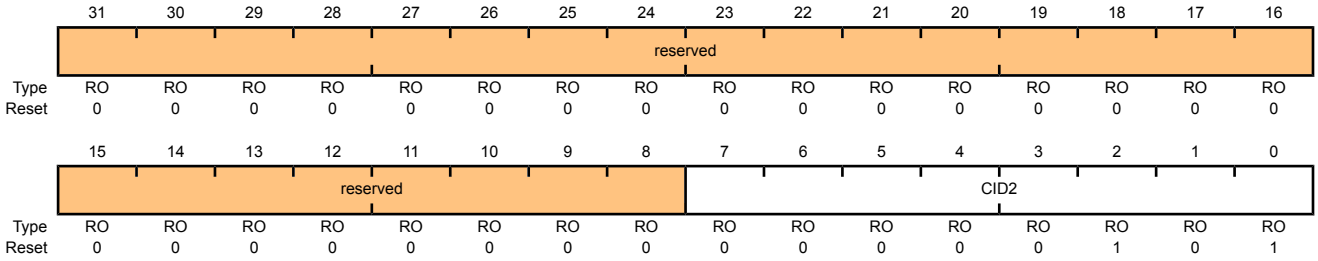
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	UART PrimeCell ID Register[15:8] Provides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

Register 25: 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



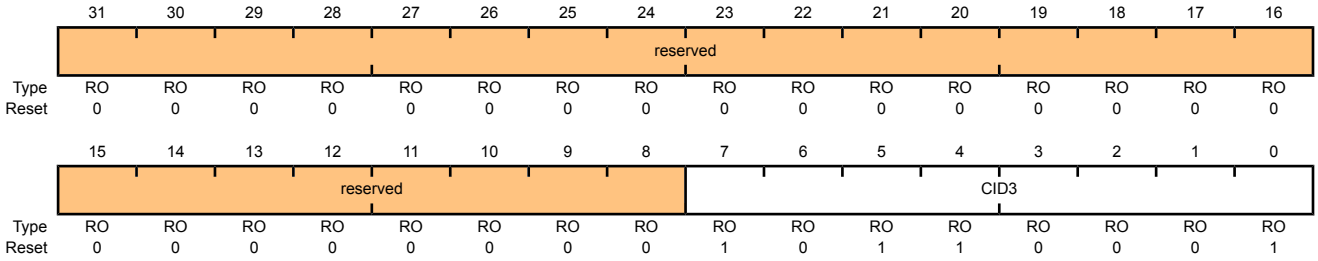
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	UART PrimeCell ID Register[23:16] Provides software a standard cross-peripheral identification system.

Register 26: 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



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.

15 Synchronous Serial Interface (SSI)

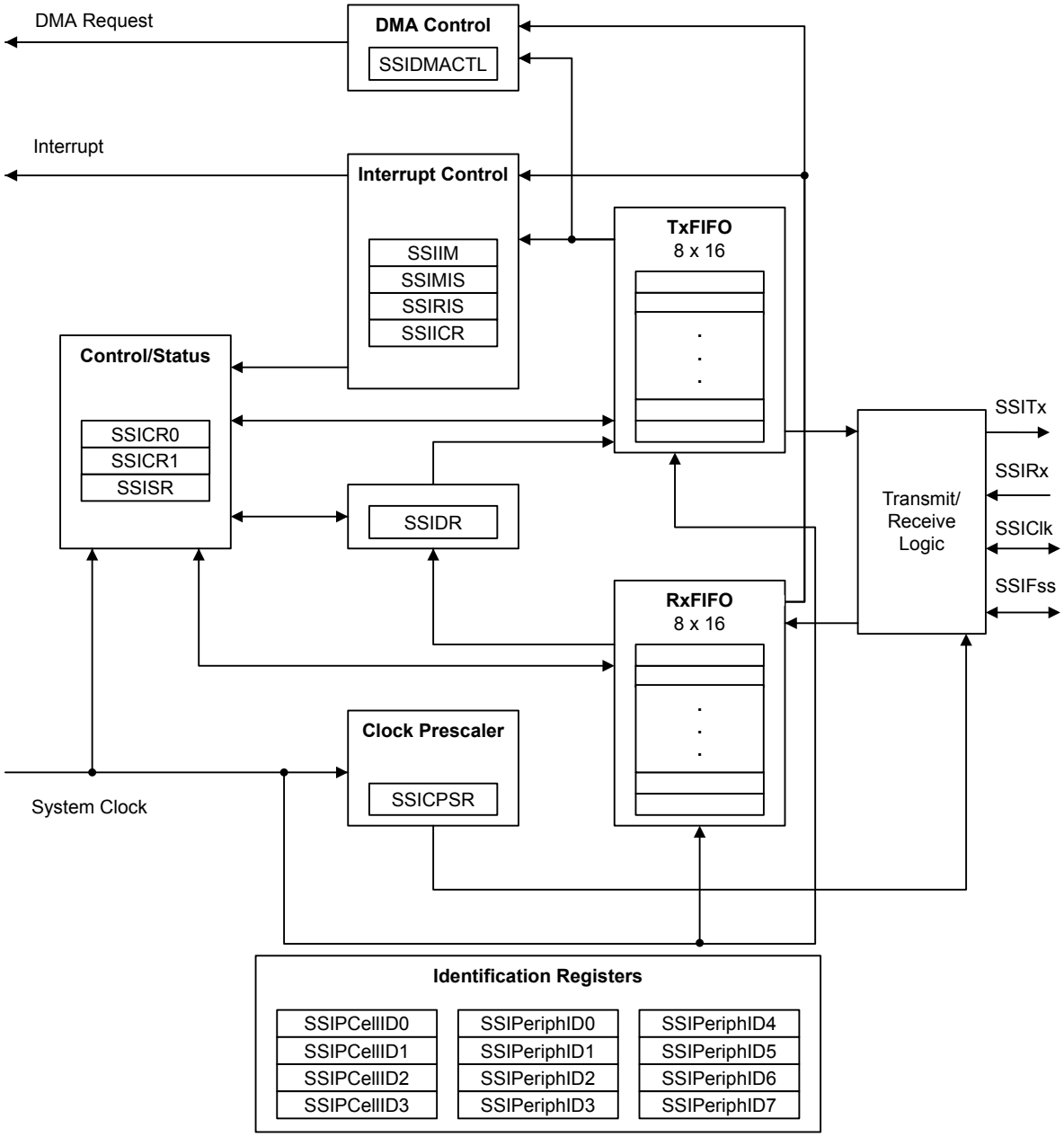
The Stellaris® microcontroller includes two Synchronous Serial Interface (SSI) modules. Each SSI is a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces.

Each Stellaris® SSI module has the following features:

- Two SSI modules, each with the following features:
- Master or slave operation
- Support for Direct Memory Access (DMA)
- 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

15.1 Block Diagram

Figure 15-1. SSI Module Block Diagram



15.2 Functional Description

The SSI performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information. The transmit and receive paths are buffered with internal FIFO memories allowing up to eight 16-bit values to be stored independently in both transmit and receive modes. The SSI also supports the DMA interface. The transmit and receive FIFOs can be programmed as destination/source addresses in the DMA module. DMA operation is enabled by setting the appropriate bit(s) in the **SSIDMACTL** register (see page 466).

15.2.1 Bit Rate Generation

The SSI includes a programmable bit rate clock divider and prescaler to generate the serial output clock. Bit rates are supported to 2 MHz and higher, although maximum bit rate is determined by peripheral devices.

The serial bit rate is derived by dividing down the input clock (FSysClk). The clock is first divided by an even prescale value CPSDVSR from 2 to 254, which is programmed in the **SSI Clock Prescale (SSICPSR)** register (see page 460). 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 453).

The frequency of the output clock SSIClk is defined by:

```
SSIClk = FSysClk / (CPSDVSR * (1 + SCR))
```

Note: 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 758 to view SSI timing parameters.

15.2.2 FIFO Operation

15.2.2.1 Transmit FIFO

The common transmit FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. The CPU writes data to the FIFO by writing the **SSI Data (SSIDR)** register (see page 457), and data is stored in the FIFO until it is read out by the transmission logic.

When configured as a master or a slave, parallel data is written into the transmit FIFO prior to serial conversion and transmission to the attached slave or master, respectively, through the SSITx pin.

15.2.2.2 Receive FIFO

The common receive FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. Received data from the serial interface is stored in the buffer until read out by the CPU, which accesses the read FIFO by reading the **SSIDR** register.

When configured as a master or slave, serial data received through the SSIRx pin is registered prior to parallel loading into the attached slave or master receive FIFO, respectively.

15.2.3 Interrupts

The SSI can generate interrupts when the following conditions are observed:

- Transmit FIFO service
- Receive FIFO service
- Receive FIFO time-out
- Receive FIFO overrun

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 461). 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 463 and page 464, respectively).

15.2.4 Frame Formats

Each data frame is between 4 and 16 bits long, depending on the size of data programmed, and is transmitted starting with the MSB. There are three basic frame types that can be selected:

- Texas Instruments synchronous serial
- Freescale SPI
- MICROWIRE

For all three formats, the serial clock (SSIClk) is held inactive while the SSI is idle, and SSIClk transitions at the programmed frequency only during active transmission or reception of data. The idle state of SSIClk is utilized to provide a receive timeout indication that occurs when the receive FIFO still contains data after a timeout period.

For Freescale SPI and MICROWIRE frame formats, the serial frame (SSIFss) pin is active Low, and is asserted (pulled down) during the entire transmission of the frame.

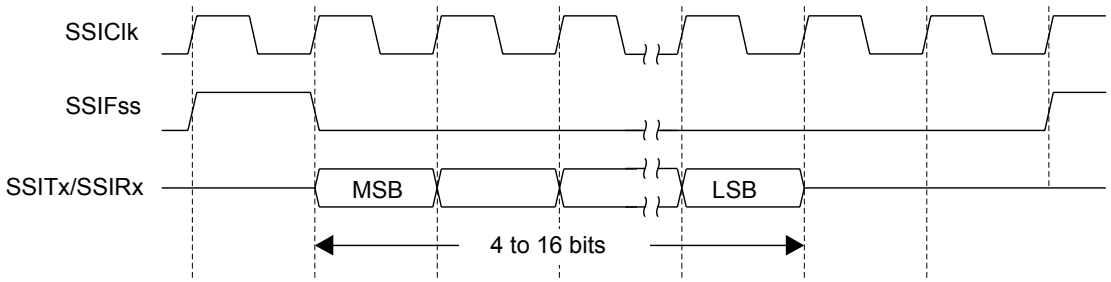
For Texas Instruments synchronous serial frame format, the SSIFss pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSI and the off-chip slave device drive their output data on the rising edge of SSIClk, and latch data from the other device on the falling edge.

Unlike the full-duplex transmission of the other two frame formats, the MICROWIRE format uses a special master-slave messaging technique, which operates at half-duplex. In this mode, when a frame begins, an 8-bit control message is transmitted to the off-chip slave. During this transmit, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the requested data. The returned data can be 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

15.2.4.1 Texas Instruments Synchronous Serial Frame Format

Figure 15-2 on page 442 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

Figure 15-2. TI Synchronous Serial Frame Format (Single Transfer)

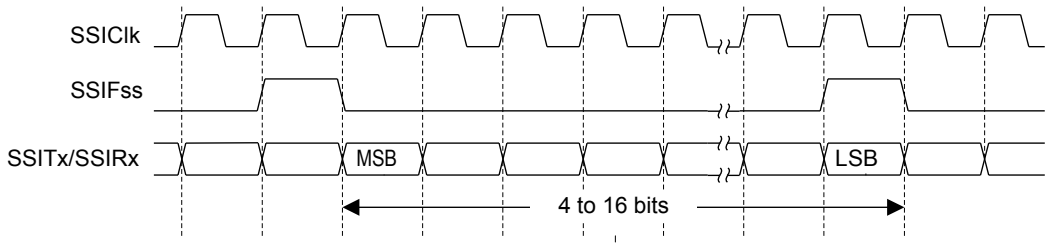


In this mode, SSIClk and SSIFss are forced Low, and the transmit data line SSITx is tristated whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, SSIFss is pulsed High for one SSIClk period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of SSIClk, the MSB of the 4 to 16-bit data frame is shifted out on the SSITx pin. Likewise, the MSB of the received data is shifted onto the SSIRx pin by the off-chip serial slave device.

Both the SSI and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each SSIClk. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SSIClk after the LSB has been latched.

Figure 15-3 on page 443 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

Figure 15-3. TI Synchronous Serial Frame Format (Continuous Transfer)



15.2.4.2 Freescale SPI Frame Format

The Freescale SPI interface is a four-wire interface where the SSIFss signal behaves as a slave select. The main feature of the Freescale SPI format is that the inactive state and phase of the SSIClk signal are programmable through the SPO and SPH bits within the **SSISCR0** control register.

SPO Clock Polarity Bit

When the SPO clock polarity control bit is Low, it produces a steady state Low value on the SSIClk pin. If the SPO bit is High, a steady state High value is placed on the SSIClk pin when data is not being transferred.

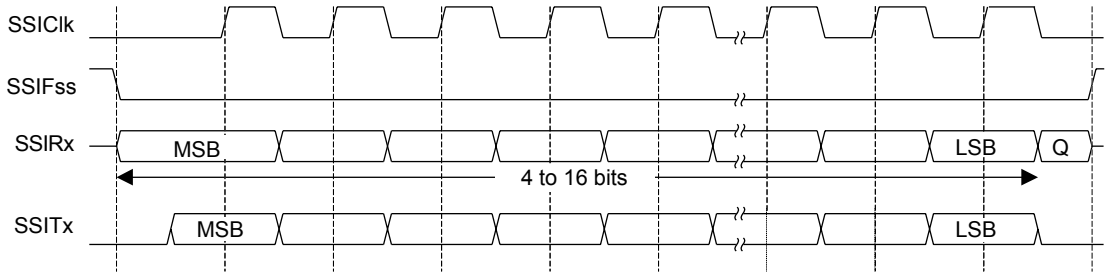
SPH Phase Control Bit

The SPH phase control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the SPH phase control bit is Low, data is captured on the first clock edge transition. If the SPH bit is High, data is captured on the second clock edge transition.

15.2.4.3 Freescale SPI Frame Format with SPO=0 and SPH=0

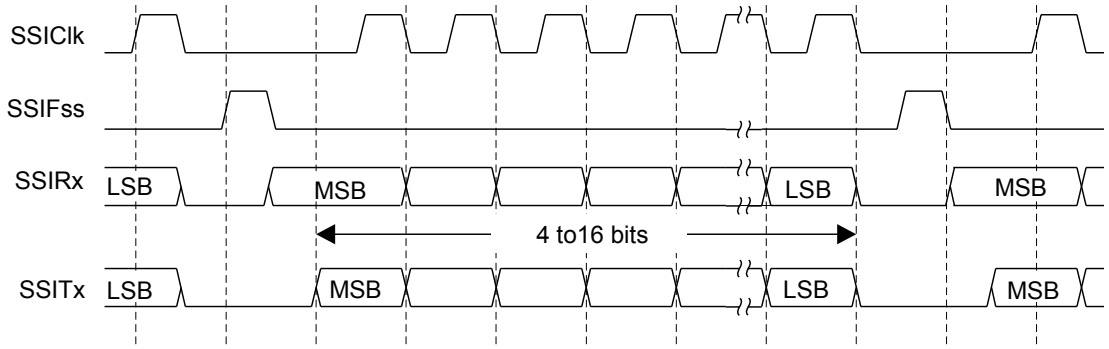
Single and continuous transmission signal sequences for Freescale SPI format with SPO=0 and SPH=0 are shown in Figure 15-4 on page 444 and Figure 15-5 on page 444.

Figure 15-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0



Note: Q is undefined.

Figure 15-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0



In this configuration, during idle periods:

- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. This causes slave data to be enabled onto the SSIRx input line of the master. The master SSITx output pad is enabled.

One half SSIClk period later, valid master data is transferred to the SSITx pin. Now that both the master and slave data have been set, the SSIClk master clock pin goes High after one further half SSIClk period.

The data is now captured on the rising and propagated on the falling edges of the SSIClk signal.

In the case of a single word transmission, after all bits of the data word have been transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

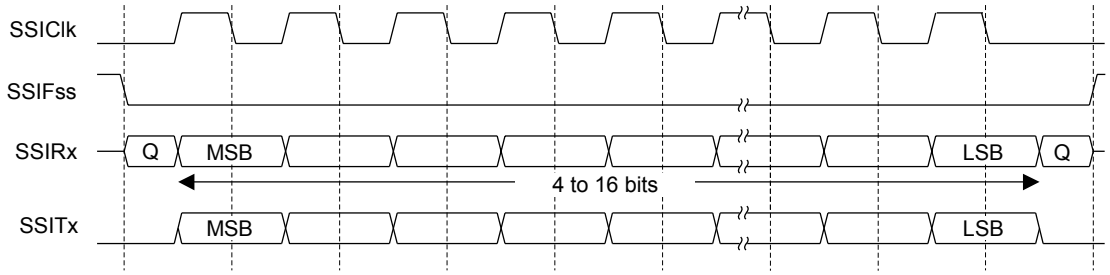
However, in the case of continuous back-to-back transmissions, the SSIFss signal must be pulsed High between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is logic zero. Therefore, the master device must raise the SSIFss pin of the slave device between each data transfer to

enable the serial peripheral data write. On completion of the continuous transfer, the SSIFss pin is returned to its idle state one SSIClk period after the last bit has been captured.

15.2.4.4 Freescale SPI Frame Format with SPO=0 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=0 and SPH=1 is shown in Figure 15-6 on page 445, which covers both single and continuous transfers.

Figure 15-6. Freescale SPI Frame Format with SPO=0 and SPH=1



Note: Q is undefined.

In this configuration, during idle periods:

- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. The master SSITx output is enabled. After a further one half SSIClk period, both master and slave valid data is enabled onto their respective transmission lines. At the same time, the SSIClk is enabled with a rising edge transition.

Data is then captured on the falling edges and propagated on the rising edges of the SSIClk signal.

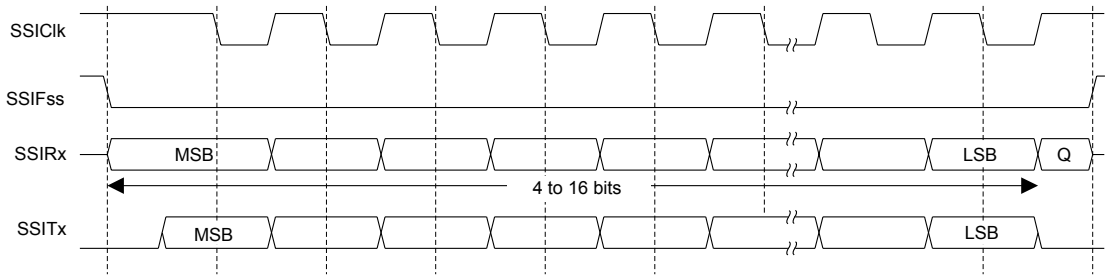
In the case of a single word transfer, after all bits have been transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

For continuous back-to-back transfers, the SSIFss pin is held Low between successive data words and termination is the same as that of the single word transfer.

15.2.4.5 Freescale SPI Frame Format with SPO=1 and SPH=0

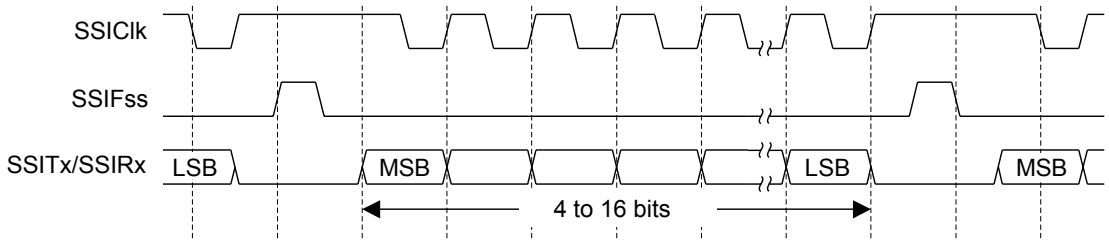
Single and continuous transmission signal sequences for Freescale SPI format with SPO=1 and SPH=0 are shown in Figure 15-7 on page 446 and Figure 15-8 on page 446.

Figure 15-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0



Note: Q is undefined.

Figure 15-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0



In this configuration, during idle periods:

- SSIClk is forced High
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low, which causes slave data to be immediately transferred onto the SSIRx line of the master. The master SSITx output pad is enabled.

One half period later, valid master data is transferred to the SSITx line. Now that both the master and slave data have been set, the SSIClk master clock pin becomes Low after one further half SSIClk period. This means that data is captured on the falling edges and propagated on the rising edges of the SSIClk signal.

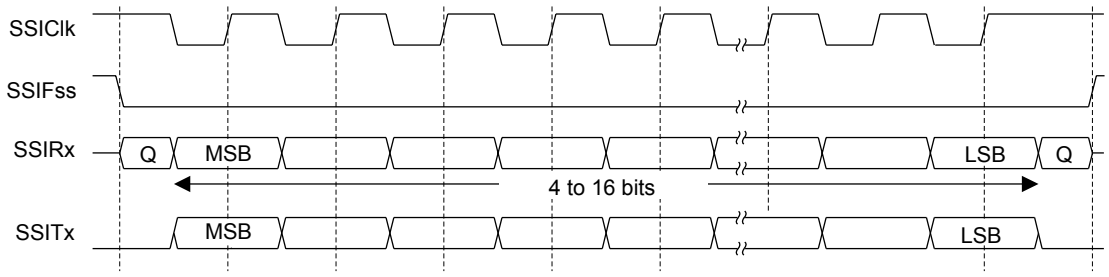
In the case of a single word transmission, after all bits of the data word are transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSIFss signal must be pulsed High between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is logic zero. Therefore, the master device must raise the SSIFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSIFss pin is returned to its idle state one SSIClk period after the last bit has been captured.

15.2.4.6 Freescale SPI Frame Format with SPO=1 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=1 and SPH=1 is shown in Figure 15-9 on page 447, which covers both single and continuous transfers.

Figure 15-9. Freescale SPI Frame Format with SPO=1 and SPH=1



Note: Q is undefined.

In this configuration, during idle periods:

- SSIClk is forced High
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and 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.

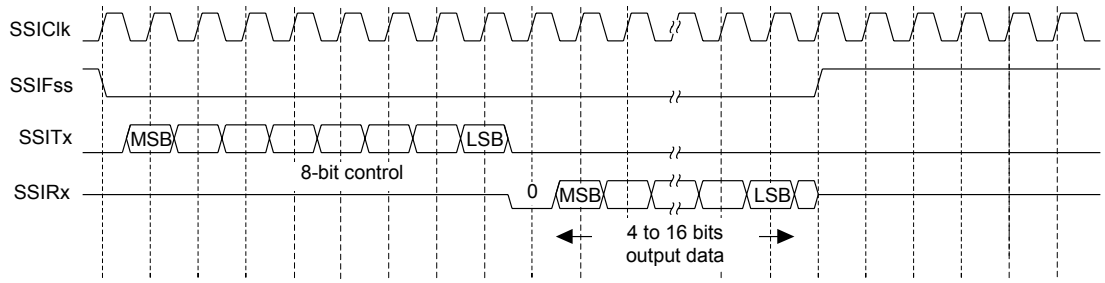
For continuous back-to-back transmissions, the SSIFss pin remains in its active Low state, until the final bit of the last word has been captured, and then returns to its idle state as described above.

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.

15.2.4.7 MICROWIRE Frame Format

Figure 15-10 on page 448 shows the MICROWIRE frame format, again for a single frame. Figure 15-11 on page 449 shows the same format when back-to-back frames are transmitted.

Figure 15-10. MICROWIRE Frame Format (Single Frame)



MICROWIRE format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex, using a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSI to the off-chip slave device. During this transmission, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

In this configuration, during idle periods:

- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low

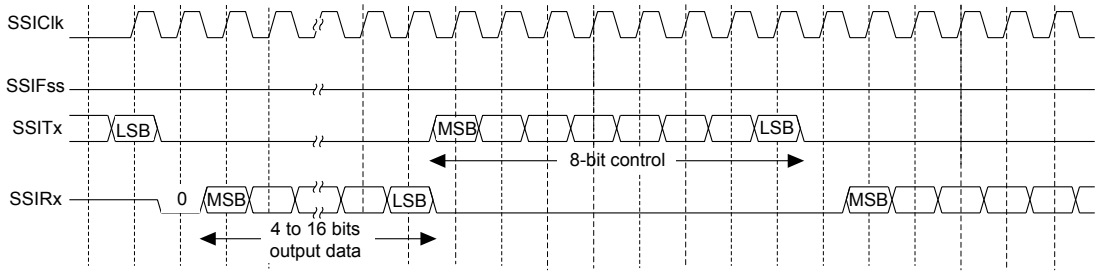
A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of SSIFss causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic, and the MSB of the 8-bit control frame to be shifted out onto the SSITx pin. SSIFss remains Low for the duration of the frame transmission. The SSIRx pin remains tristated during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on the rising edge of each SSIClk. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state, and the slave responds by transmitting data back to the SSI. Each bit is driven onto the SSIRx line on the falling edge of SSIClk. The SSI in turn latches each bit on the rising edge of SSIClk. At the end of the frame, for single transfers, the SSIFss signal is pulled High one clock period after the last bit has been latched in the receive serial shifter, which causes the data to be transferred to the receive FIFO.

Note: The off-chip slave device can tristate the receive line either on the falling edge of SSIClk after the LSB has been latched by the receive shifter, or when the SSIFss pin goes High.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the SSIFss line is continuously asserted (held Low) and transmission of data occurs back-to-back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge of SSIClk, after the LSB of the frame has been latched into the SSI.

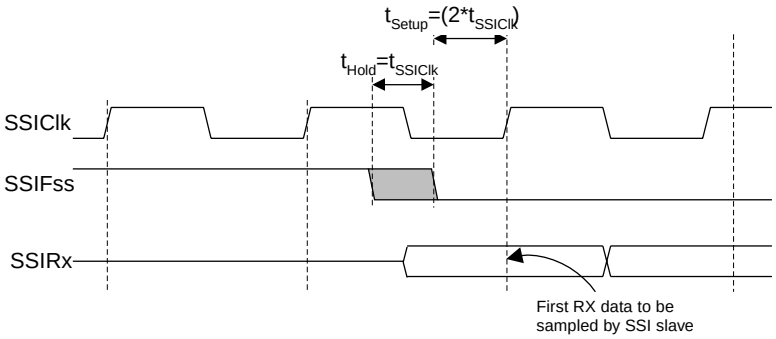
Figure 15-11. MICROWIRE Frame Format (Continuous Transfer)



In the MICROWIRE mode, the SSI slave samples the first bit of receive data on the rising edge of SSIClk after SSIFss has gone Low. Masters that drive a free-running SSIClk must ensure that the SSIFss signal has sufficient setup and hold margins with respect to the rising edge of SSIClk.

Figure 15-12 on page 449 illustrates these setup and hold time requirements. With respect to the SSIClk rising edge on which the first bit of receive data is to be sampled by the SSI slave, SSIFss must have a setup of at least two times the period of SSIClk on which the SSI operates. With respect to the SSIClk rising edge previous to this edge, SSIFss must have a hold of at least one SSIClk period.

Figure 15-12. MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements



15.2.5 DMA Operation

The SSI peripheral provides an interface connected to the µDMA controller. The DMA operation of the SSI is enabled through the **SSI DMA Control (SSIDMACTL)** register. When DMA operation is enabled, the SSI will assert a DMA request on the receive or transmit channel when the associated FIFO can transfer data. For the receive channel, a single transfer request is asserted whenever there is any data in the receive FIFO. A burst transfer request is asserted whenever the amount of data in the receive FIFO is 4 or more items. For the transmit channel, a single transfer request is asserted whenever there is at least one empty location in the transmit FIFO. The burst request is asserted whenever the transmit FIFO has 4 or more empty slots. The single and burst DMA transfer requests are handled automatically by the µDMA controller depending how the DMA channel is configured. To enable DMA operation for the receive channel, the RXDMAE bit of the **DMA Control (SSIDMACTL)** register should be set. To enable DMA operation for the transmit channel, the TXDMAE bit of **SSIDMACTL** should be set. If DMA is enabled, then the µDMA controller will trigger an interrupt when a transfer is complete. The interrupt will occur on the SSI interrupt vector. Therefore, if interrupts are used for SSI operation and DMA is enabled, the SSI interrupt handler must be designed to handle the µDMA completion interrupt.

See “Micro Direct Memory Access (µDMA)” on page 194 for more details about programming the µDMA controller.

15.3 Initialization and Configuration

To use the SSI, its peripheral clock must be enabled by setting the SSI bit in the **RCGC1** register.

For each of the frame formats, the SSI is configured using the following steps:

1. Ensure that the SSE bit in the **SSICR1** register is disabled before making any configuration changes.
2. Select whether the SSI is a master or slave:
   a. For master operations, set the **SSICR1** register to 0x0000.0000.
   b. For slave mode (output enabled), set the **SSICR1** register to 0x0000.0004.
   c. For slave mode (output disabled), set the **SSICR1** register to 0x0000.000C.
3. Configure the clock prescale divisor by writing the **SSICPSR** register.
4. Write the **SSICR0** register with the following configuration:
   - Serial clock rate (SCR)
   - Desired clock phase/polarity, if using Freescale SPI mode (SPH and SPO)
   - The protocol mode: Freescale SPI, TI SSF, MICROWIRE (FRF)
   - The data size (DSS)
5. Optionally, configure the µDMA channel (see “Micro Direct Memory Access (µDMA)” on page 194) and enable the DMA option(s) in the **SSIDMACTL** register.
6. Enable the SSI by setting the SSE bit in the **SSICR1** register.

As an example, assume the SSI must be configured to operate with the following parameters:

- Master operation
- Freescale SPI mode (SPO=1, SPH=1)
- 1 Mbps bit rate
- 8 data bits

Assuming the system clock is 20 MHz, the bit rate calculation would be:

```
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.

15.4 Register Map

Table 15-1 on page 451 lists the SSI registers. The offset listed is a hexadecimal increment to the register's address, relative to that SSI module's base address:

- SSI0: 0x4000.8000
- SSI1: 0x4000.9000

Note: The SSI must be disabled (see the SSE bit in the **SSICR1** register) before any of the control registers are reprogrammed.

Table 15-1. SSI Register Map

Offset	Name	Type	Reset	Description	See page
0x000	SSICR0	R/W	0x0000.0000	SSI Control 0	453
0x004	SSICR1	R/W	0x0000.0000	SSI Control 1	455
0x008	SSIDR	R/W	0x0000.0000	SSI Data	457
0x00C	SSISR	RO	0x0000.0003	SSI Status	458
0x010	SSICPSR	R/W	0x0000.0000	SSI Clock Prescale	460
0x014	SSIIM	R/W	0x0000.0000	SSI Interrupt Mask	461
0x018	SSIRIS	RO	0x0000.0008	SSI Raw Interrupt Status	463
0x01C	SSIMIS	RO	0x0000.0000	SSI Masked Interrupt Status	464
0x020	SSIICR	W1C	0x0000.0000	SSI Interrupt Clear	465
0x024	SSIDMACTL	R/W	0x0000.0000	SSI DMA Control	466
0xFD0	SSIPeriphID4	RO	0x0000.0000	SSI Peripheral Identification 4	467
0xFD4	SSIPeriphID5	RO	0x0000.0000	SSI Peripheral Identification 5	468
0xFD8	SSIPeriphID6	RO	0x0000.0000	SSI Peripheral Identification 6	469
0xFDC	SSIPeriphID7	RO	0x0000.0000	SSI Peripheral Identification 7	470
0xFE0	SSIPeriphID0	RO	0x0000.0022	SSI Peripheral Identification 0	471
0xFE4	SSIPeriphID1	RO	0x0000.0000	SSI Peripheral Identification 1	472
0xFE8	SSIPeriphID2	RO	0x0000.0018	SSI Peripheral Identification 2	473
0xFEC	SSIPeriphID3	RO	0x0000.0001	SSI Peripheral Identification 3	474
0xFF0	SSIPCellID0	RO	0x0000.000D	SSI PrimeCell Identification 0	475
0xFF4	SSIPCellID1	RO	0x0000.00F0	SSI PrimeCell Identification 1	476

Offset	Name	Type	Reset	Description	See page
0xFF8	SSIPCellID2	RO	0x0000.0005	SSI PrimeCell Identification 2	477
0xFFC	SSIPCellID3	RO	0x0000.00B1	SSI PrimeCell Identification 3	478

15.5 Register Descriptions

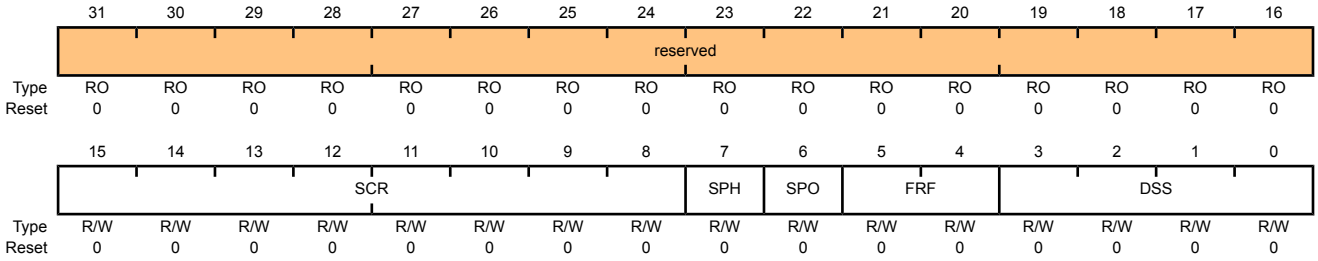
The remainder of this section lists and describes the SSI registers, in numerical order by address offset.

Register 1: SSI Control 0 (SSICR0), offset 0x000

SSICR0 is control register 0 and contains bit fields that control various functions within the SSI module. Functionality such as protocol mode, clock rate, and data size are configured in this register.

SSI Control 0 (SSICR0)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x000
Type R/W, reset 0x0000.0000



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.

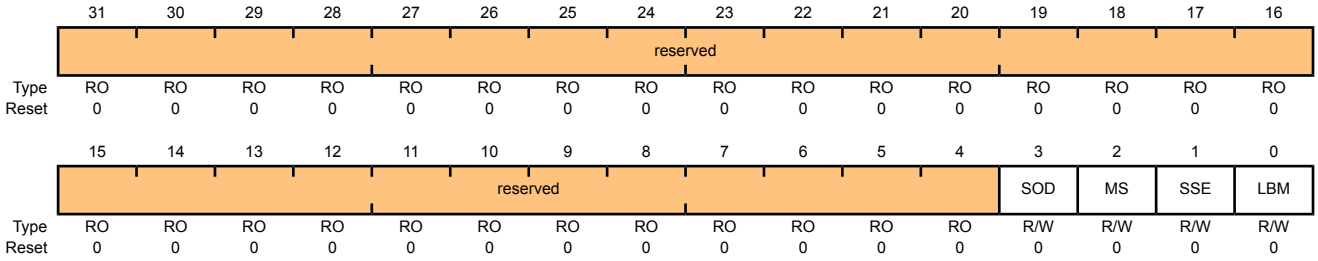
Bit/Field	Name	Type	Reset	Description
5:4	FRF	R/W	0x0	SSI Frame Format Select The FRF values are defined as follows: Value Frame Format 0x0 Freescale SPI Frame Format 0x1 Texas Instruments Synchronous Serial Frame Format 0x2 MICROWIRE Frame Format 0x3 Reserved
3:0	DSS	R/W	0x00	SSI Data Size Select The DSS values are defined as follows: Value Data Size 0x0-0x2 Reserved 0x3 4-bit data 0x4 5-bit data 0x5 6-bit data 0x6 7-bit data 0x7 8-bit data 0x8 9-bit data 0x9 10-bit data 0xA 11-bit data 0xB 12-bit data 0xC 13-bit data 0xD 14-bit data 0xE 15-bit data 0xF 16-bit data

Register 2: SSI Control 1 (SSICR1), offset 0x004

SSICR1 is control register 1 and contains bit fields that control various functions within the SSI module. Master and slave mode functionality is controlled by this register.

SSI Control 1 (SSICR1)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x004
Type R/W, reset 0x0000.0000



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).

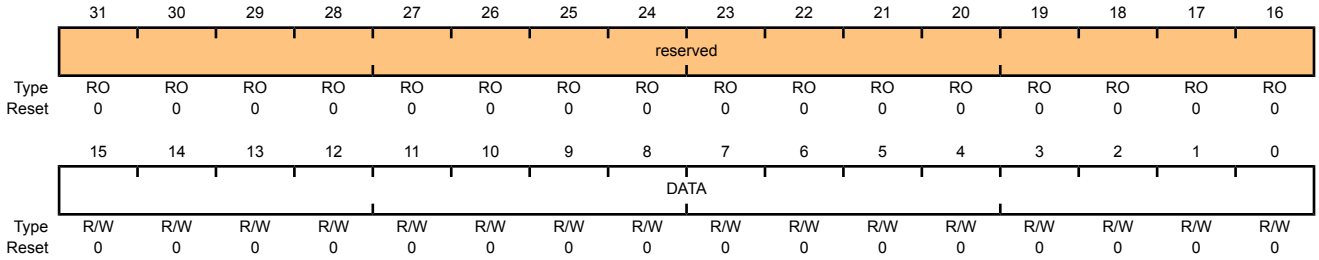
When **SSIDR** is written to, the entry in the transmit FIFO (pointed to by the write pointer) is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. It is loaded into the transmit serial shifter, then serially shifted out onto the SSITx pin at the programmed 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
SSI1 base: 0x4000.9000
Offset 0x008
Type R/W, reset 0x0000.0000



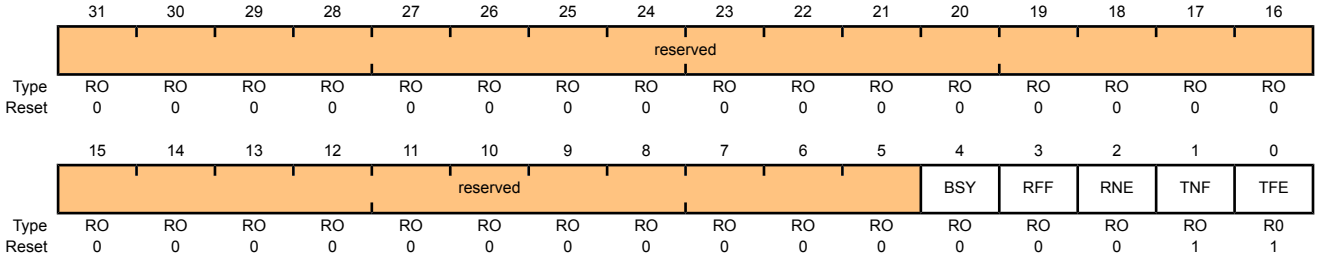
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DATA	R/W	0x0000	SSI Receive/Transmit Data A read operation reads the receive FIFO. A write operation writes the transmit FIFO. Software must right-justify data when the SSI is programmed for a data size that is less than 16 bits. Unused bits at the top are ignored by the transmit logic. The receive logic automatically right-justifies the data.

Register 4: SSI Status (SSISR), offset 0x00C

SSISR is a status register that contains bits that indicate the FIFO fill status and the SSI busy status.

SSI Status (SSISR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x00C
Type RO, reset 0x0000.0003



Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	BSY	RO	0	SSI Busy Bit The BSY values are defined as follows: Value Description 0 SSI is idle. 1 SSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.
3	RFF	RO	0	SSI Receive FIFO Full The RFF values are defined as follows: Value Description 0 Receive FIFO is not full. 1 Receive FIFO is full.
2	RNE	RO	0	SSI Receive FIFO Not Empty The RNE values are defined as follows: Value Description 0 Receive FIFO is empty. 1 Receive FIFO is not empty.

Bit/Field	Name	Type	Reset	Description
1	TNF	RO	1	SSI Transmit FIFO Not Full The TNF values are defined as follows: Value Description 0 Transmit FIFO is full. 1 Transmit FIFO is not full.
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.

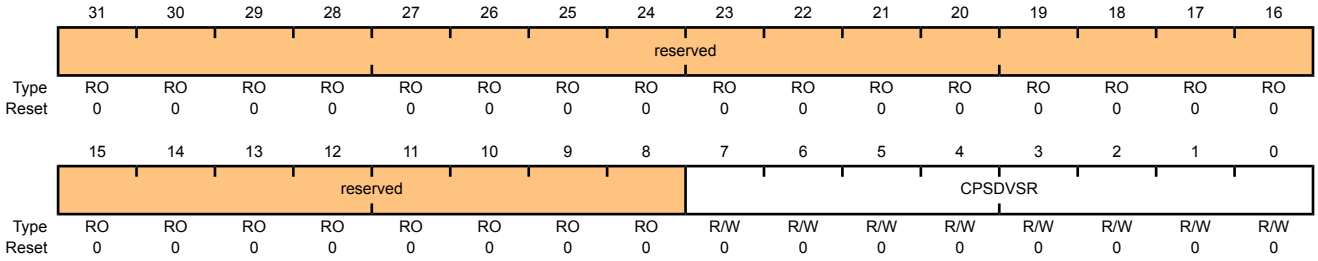
Register 5: SSI Clock Prescale (SSICPSR), offset 0x010

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
SSI1 base: 0x4000.9000
Offset 0x010
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CPSDVSR	R/W	0x00	SSI Clock Prescale Divisor This value must be an even number from 2 to 254, depending on the frequency of SSIClk. The LSB always returns 0 on reads.

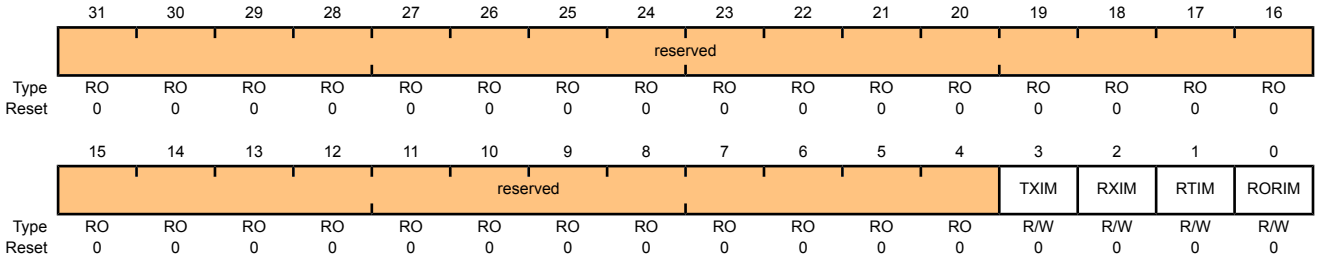
Register 6: SSI Interrupt Mask (SSIIM), offset 0x014

The **SSIIM** register is the interrupt mask set or clear register. It is a read/write register and all bits are cleared to 0 on reset.

On a read, this register gives the current value of the mask on the relevant interrupt. A write of 1 to the particular bit sets the mask, enabling the interrupt to be read. A write of 0 clears the corresponding mask.

SSI Interrupt Mask (SSIIM)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x014
Type R/W, reset 0x0000.0000



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 `RXIM` 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.

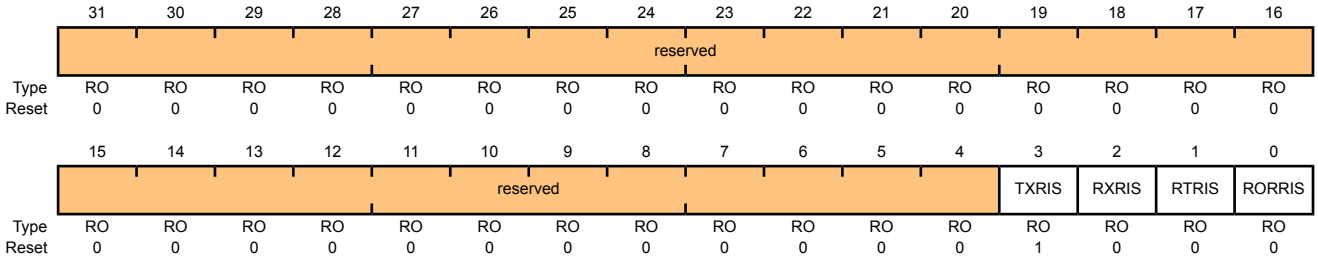
Bit/Field	Name	Type	Reset	Description
0	RORIM	R/W	0	SSI Receive Overrun Interrupt Mask The RORIM values are defined as follows: Value Description 0 RX FIFO overrun interrupt is masked. 1 RX FIFO overrun interrupt is not masked.

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
SSI1 base: 0x4000.9000
Offset 0x018
Type RO, reset 0x0000.0008



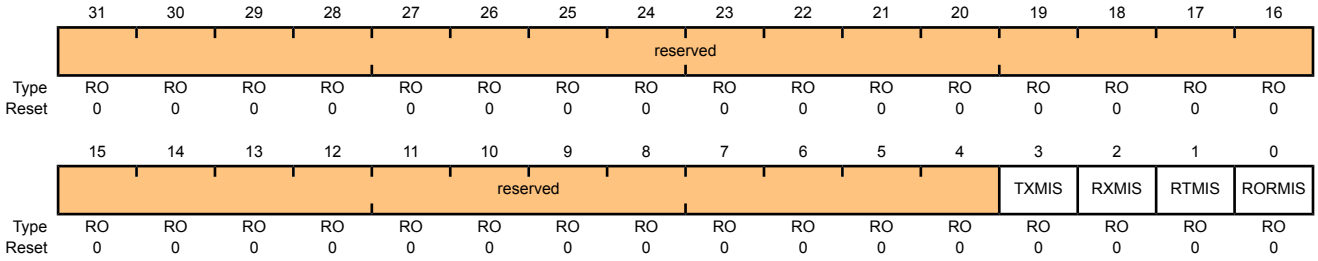
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXRIS	RO	1	SSI Transmit FIFO Raw Interrupt Status 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
SSI1 base: 0x4000.9000
Offset 0x01C
Type RO, reset 0x0000.0000



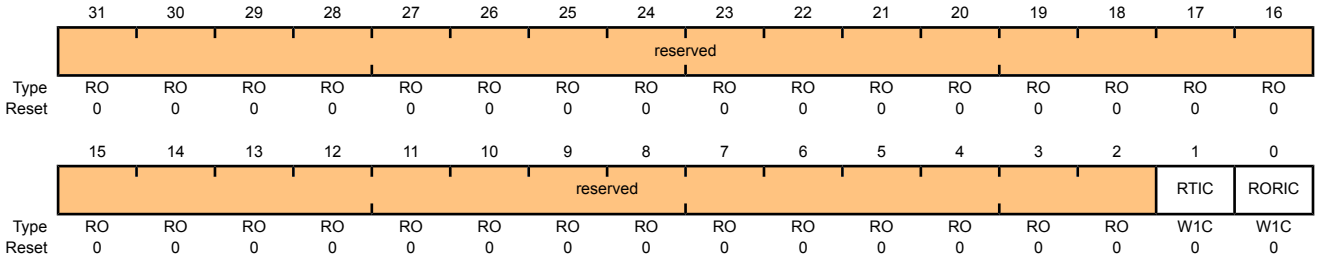
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	TXMIS	RO	0	SSI Transmit FIFO Masked Interrupt Status 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
SSI1 base: 0x4000.9000
Offset 0x020
Type W1C, reset 0x0000.0000



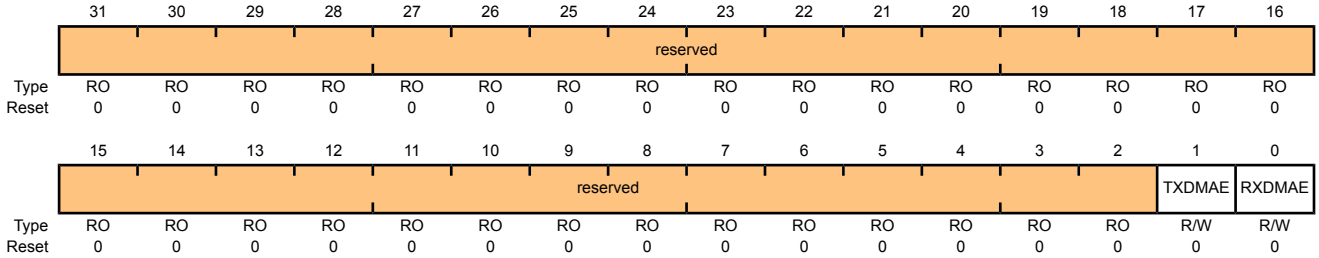
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	RTIC	W1C	0	SSI Receive Time-Out Interrupt Clear The `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 DMA Control (SSIDMACTL), offset 0x024

The **SSIDMACTL** register is the DMA control register.

SSI DMA Control (SSIDMACTL)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0x024
Type R/W, reset 0x0000.0000



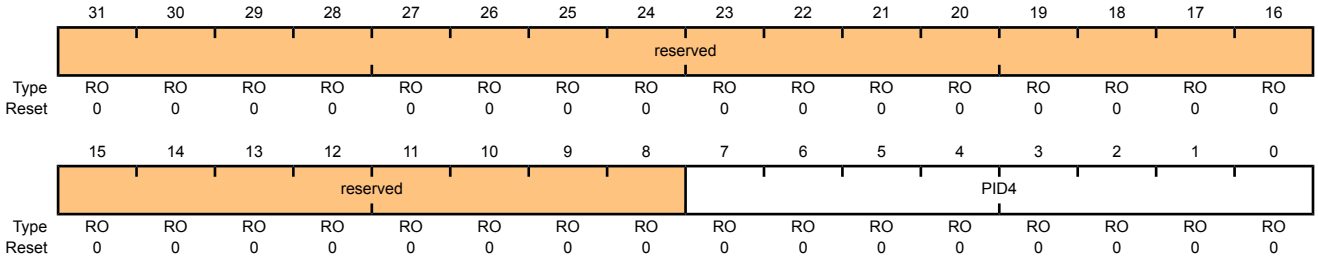
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	TXDMAE	R/W	0	Transmit DMA Enable If this bit is set to 1, DMA for the transmit FIFO is enabled.
0	RXDMAE	R/W	0	Receive DMA Enable If this bit is set to 1, DMA for the receive FIFO is enabled.

Register 11: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 4 (SSIPeriphID4)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFD0
Type RO, reset 0x0000.0000



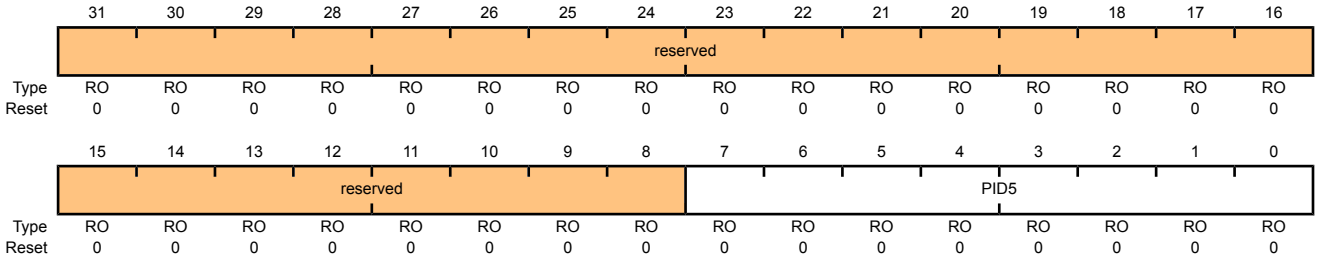
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID4	RO	0x00	SSI Peripheral ID Register[7:0] Can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

Register 12: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 5 (SSIPeriphID5)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFD4
Type RO, reset 0x0000.0000



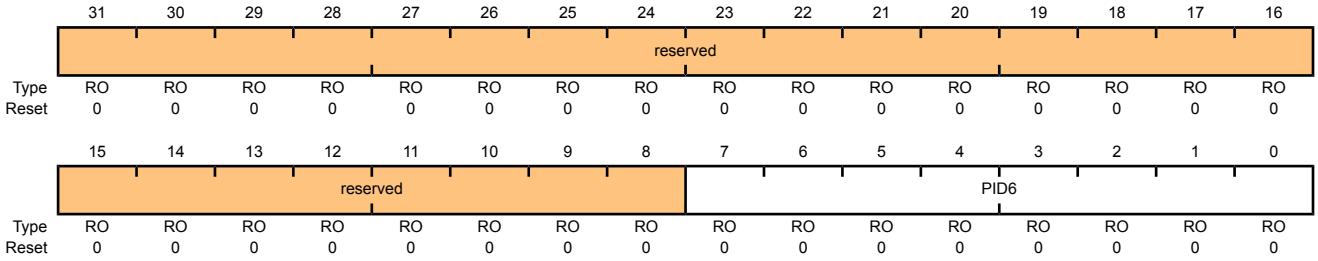
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID5	RO	0x00	SSI Peripheral ID Register[15:8] Can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

Register 13: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 6 (SSIPeriphID6)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFD8
Type RO, reset 0x0000.0000



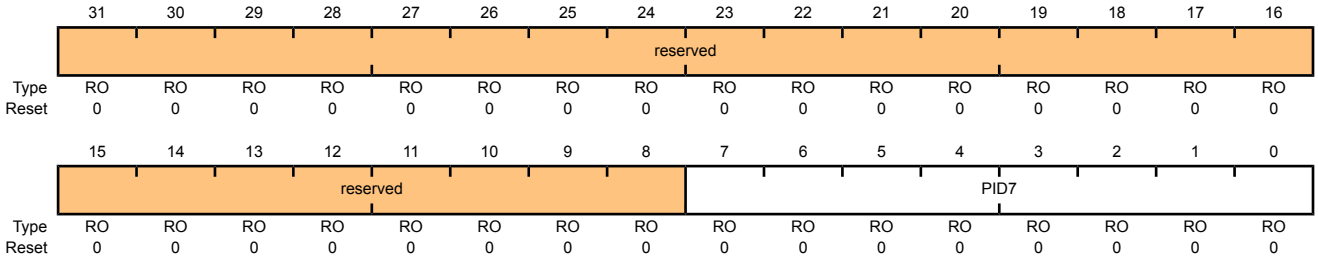
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID6	RO	0x00	SSI Peripheral ID Register[23:16]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

Register 14: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 7 (SSIPeriphID7)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFDC
Type RO, reset 0x0000.0000



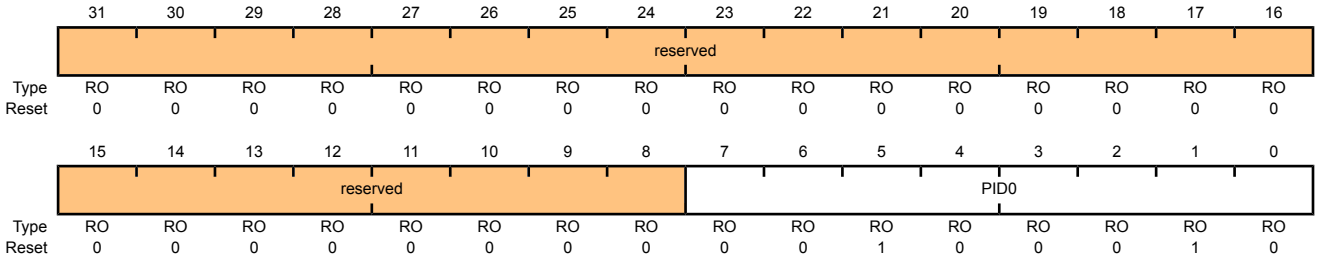
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID7	RO	0x00	SSI Peripheral ID Register[31:24]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

Register 15: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 0 (SSIPeriphID0)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFE0
Type RO, reset 0x0000.0022



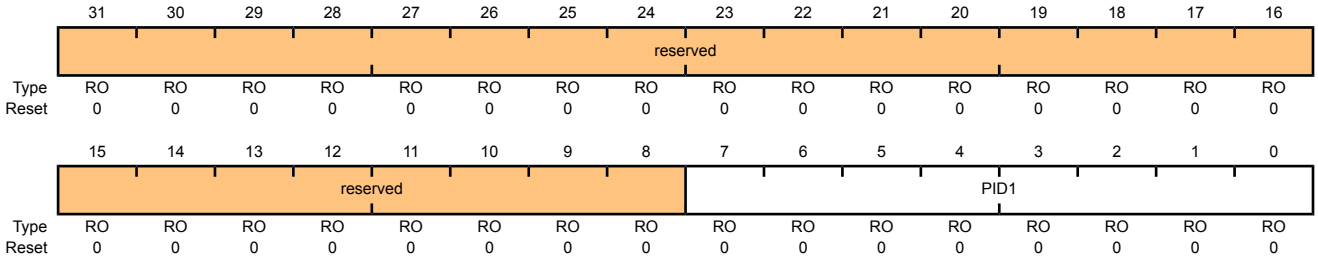
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID0	RO	0x22	SSI Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 16: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 1 (SSIPeriphID1)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFE4
Type RO, reset 0x0000.0000



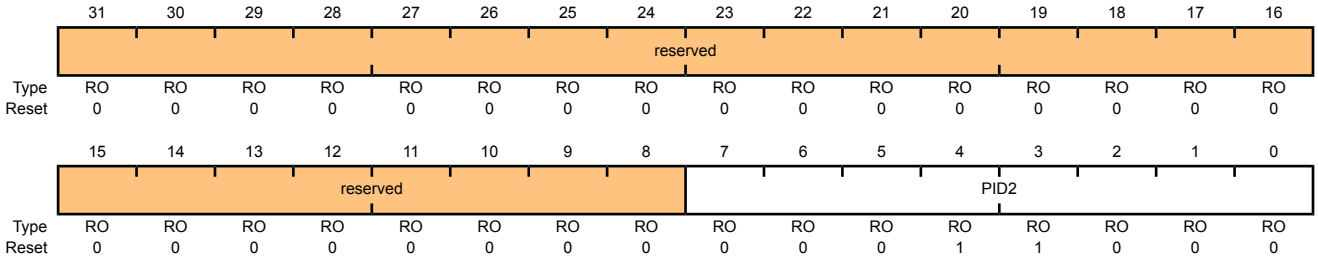
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID1	RO	0x00	SSI Peripheral ID Register [15:8]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

Register 17: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 2 (SSIPeriphID2)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFE8
Type RO, reset 0x0000.0018



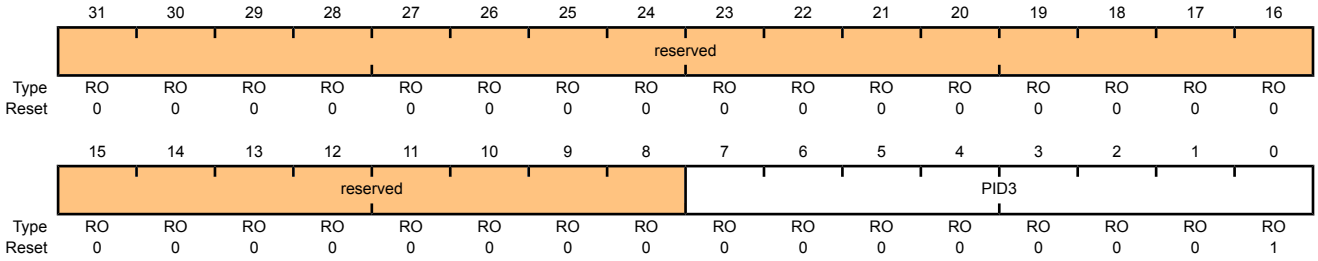
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID2	RO	0x18	SSI Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 18: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 3 (SSIPeriphID3)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFEC
Type RO, reset 0x0000.0001



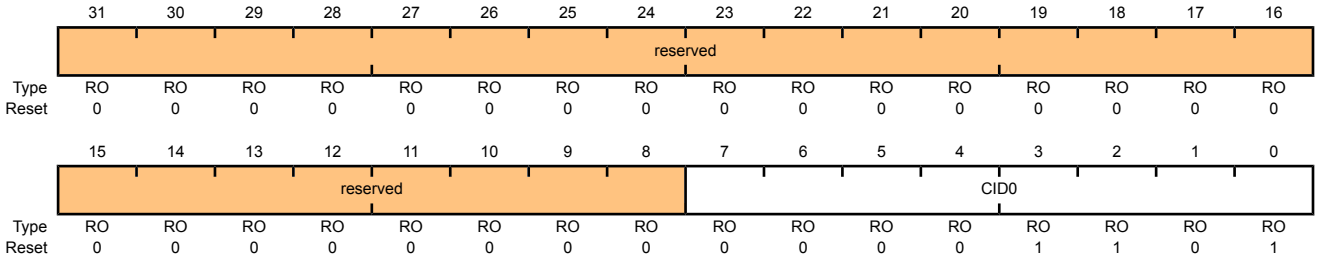
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	PID3	RO	0x01	SSI Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 19: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0

The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

SSI PrimeCell Identification 0 (SSIPCellID0)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFF0
Type RO, reset 0x0000.000D



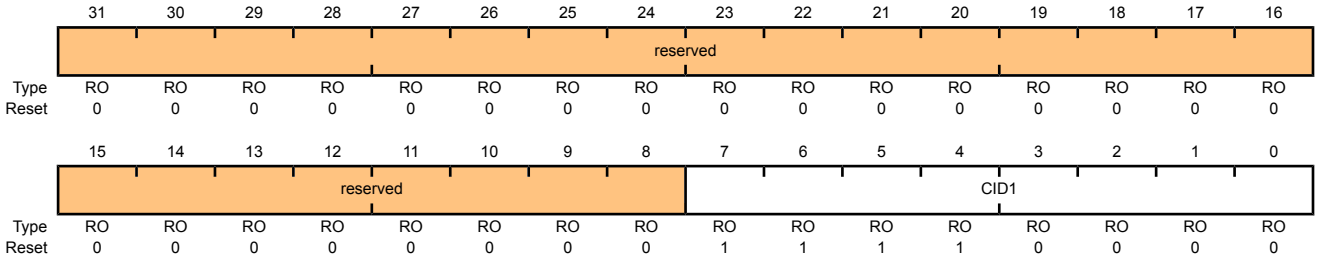
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID0	RO	0x0D	SSI PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

Register 20: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4

The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

SSI PrimeCell Identification 1 (SSIPCellID1)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFF4
Type RO, reset 0x0000.00F0



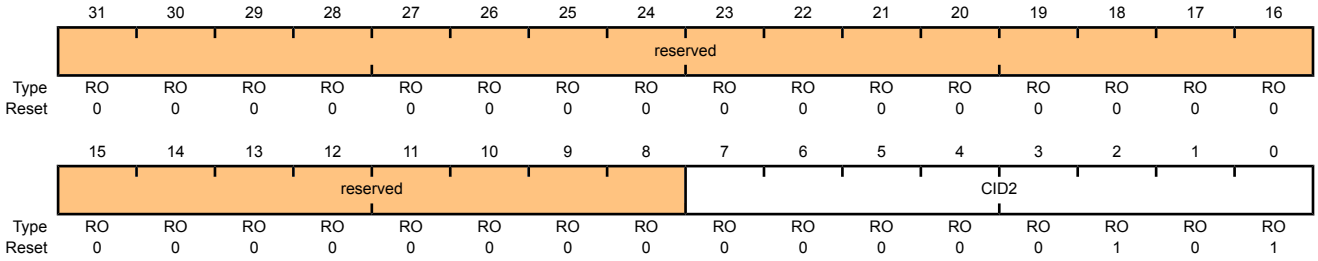
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID1	RO	0xF0	SSI PrimeCell ID Register [15:8] Provides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

Register 21: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8

The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

SSI PrimeCell Identification 2 (SSIPCellID2)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFF8
Type RO, reset 0x0000.0005



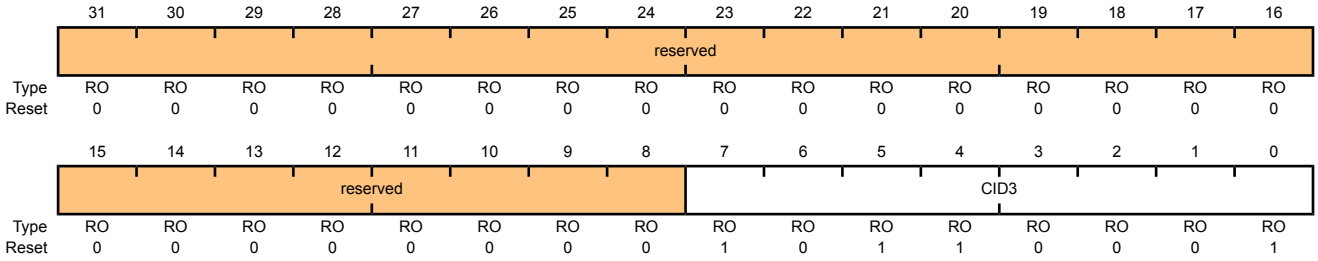
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID2	RO	0x05	SSI PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

Register 22: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC

The **SSIPCellIDn** registers are hard-coded, and the fields within the register determine the reset value.

SSI PrimeCell Identification 3 (SSIPCellID3)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
Offset 0xFFC
Type RO, reset 0x0000.00B1



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	CID3	RO	0xB1	SSI PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

16 Inter-Integrated Circuit (I^2C) Interface

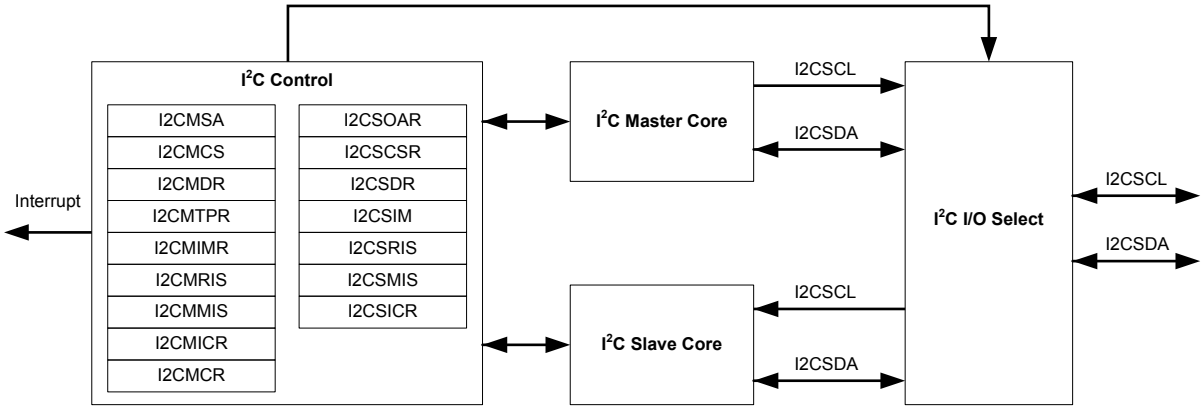
The Inter-Integrated Circuit (I^2C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDA and a serial clock line SCL), and interfaces to external I^2C devices such as serial memory (RAMs and ROMs), networking devices, LCDs, tone generators, and so on. The I^2C bus may also be used for system testing and diagnostic purposes in product development and manufacture. The LM3S5749 microcontroller includes two I^2C modules, providing the ability to interact (both send and receive) with other I^2C devices on the bus.

The Stellaris® I^2C interface has the following features:

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

16.1 Block Diagram

Figure 16-1. I2C Block Diagram

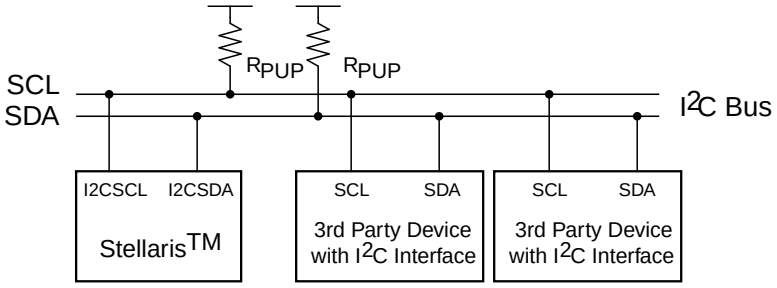


16.2 Functional Description

Each I2C module is comprised of both master and slave functions which are implemented as separate peripherals. For proper operation, the SDA and SCL pins must be connected to bi-directional open-drain pads. A typical I2C bus configuration is shown in Figure 16-2 on page 480.

See "Inter-Integrated Circuit (I2C) Interface" on page 760 for I2C timing diagrams.

Figure 16-2. I2C Bus Configuration



16.2.1 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 481) is unrestricted, but each byte has to be followed by an acknowledge bit, and data must be transferred MSB first. When a receiver cannot receive another complete byte, it can hold the clock line SCL Low and force the transmitter into a wait state. The data transfer continues when the receiver releases the clock SCL.

16.2.1.1 START and STOP Conditions

The protocol of the I^2C bus defines two states to begin and end a transaction: START and STOP. A High-to-Low transition on the SDA line while the SCL is High is defined as a START condition, and a Low-to-High transition on the SDA line while SCL is High is defined as a STOP condition. The bus is considered busy after a START condition and free after a STOP condition. See Figure 16-3 on page 481.

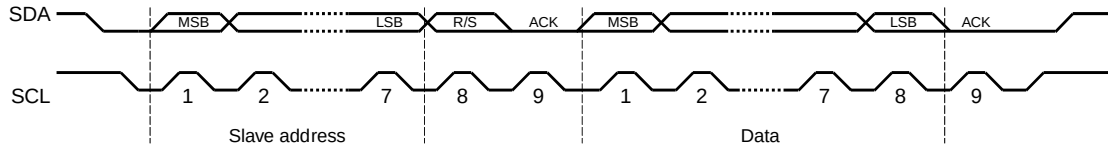
Figure 16-3. START and STOP Conditions

When operating in slave mode, two bits in the **I2CSRIS** register indicate detection of start and stop conditions on the bus; while two bits in the **I2CSMIS** register allow start and stop conditions to be promoted to controller interrupts (when interrupts are enabled).

16.2.1.2 Data Format with 7-Bit Address

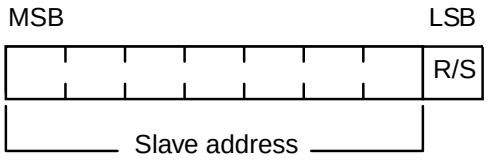
Data transfers follow the format shown in Figure 16-4 on page 481. After the START condition, a slave address is sent. This address is 7-bits long followed by an eighth bit, which is a data direction bit (R/S bit in the **I2CMSA** register). A zero indicates a transmit operation (send), and a one indicates a request for data (receive). A data transfer is always terminated by a STOP condition generated by the master, however, a master can initiate communications with another device on the bus by generating a repeated START condition and addressing another slave without first generating a STOP condition. Various combinations of receive/send formats are then possible within a single transfer.

Figure 16-4. Complete Data Transfer with a 7-Bit Address



The first seven bits of the first byte make up the slave address (see Figure 16-5 on page 481). The eighth bit determines the direction of the message. A zero in the R/S position of the first byte means that the master will write (send) data to the selected slave, and a one in this position means that the master will receive data from the slave.

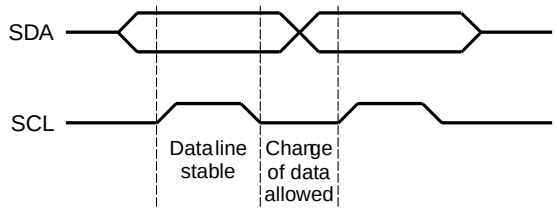
Figure 16-5. R/S Bit in First Byte



16.2.1.3 Data Validity

The data on the SDA line must be stable during the high period of the clock, and the data line can only change when SCL is Low (see Figure 16-6 on page 482).

Figure 16-6. Data Validity During Bit Transfer on the I2C Bus



16.2.1.4 Acknowledge

All bus transactions have a required acknowledge clock cycle that is generated by the master. During the acknowledge cycle, the transmitter (which can be the master or slave) releases the SDA line. To acknowledge the transaction, the receiver must pull down SDA during the acknowledge clock cycle. The data sent out by the receiver during the acknowledge cycle must comply with the data validity requirements described in “Data Validity” on page 482.

When a slave receiver does not acknowledge the slave address, SDA must be left High by the slave so that the master can generate a STOP condition and abort the current transfer. If the master device is acting as a receiver during a transfer, it is responsible for acknowledging each transfer made by the slave. Since the master controls the number of bytes in the transfer, it signals the end of data to the slave transmitter by not generating an acknowledge on the last data byte. The slave transmitter must then release SDA to allow the master to generate the STOP or a repeated START condition.

16.2.1.5 Arbitration

A master may start a transfer only if the bus is idle. It's possible for two or more masters to generate a START condition within minimum hold time of the START condition. In these situations, an arbitration scheme takes place on the SDA line, while SCL is High. During arbitration, the first of the competing master devices to place a '1' (High) on SDA while another master transmits a '0' (Low) will switch off its data output stage and retire until the bus is idle again.

Arbitration can take place over several bits. Its first stage is a comparison of address bits, and if both masters are trying to address the same device, arbitration continues on to the comparison of data bits.

16.2.2 Available Speed Modes

The I2C clock rate is determined by the parameters: `CLK_PRD`, `TIMER_PRD`, `SCL_LP`, and `SCL_HP`.

where:

`CLK_PRD` is the system clock period

`SCL_LP` is the low phase of SCL (fixed at 6)

`SCL_HP` is the high phase of SCL (fixed at 4)

`TIMER_PRD` is the programmed value in the **I2C Master Timer Period (I2CMTPR)** register (see page 500).

The I2C clock period is calculated as follows:

```
SCL_PERIOD = 2*(1 + TIMER_PRD)*(SCL_LP + SCL_HP)*CLK_PRD
```

For example:

```
CLK_PRD = 50 ns
TIMER_PRD = 2
SCL_LP=6
SCL_HP=4
```

yields a SCL frequency of:

```
1/T = 333 Khz
```

Table 16-1 on page 483 gives examples of timer period, system clock, and speed mode (Standard or Fast).

Table 16-1. Examples of I²C Master Timer Period versus Speed Mode

System Clock	Timer Period	Standard Mode	Timer Period	Fast Mode
4 MHz	0x01	100 Kbps	-	-
6 MHz	0x02	100 Kbps	-	-
12.5 MHz	0x06	89 Kbps	0x01	312 Kbps
16.7 MHz	0x08	93 Kbps	0x02	278 Kbps
20 MHz	0x09	100 Kbps	0x02	333 Kbps
25 MHz	0x0C	96.2 Kbps	0x03	312 Kbps
33 MHz	0x10	97.1 Kbps	0x04	330 Kbps
40 MHz	0x13	100 Kbps	0x04	400 Kbps
50 MHz	0x18	100 Kbps	0x06	357 Kbps

16.2.3 Interrupts

The I²C can generate interrupts when the following conditions are observed:

- Master transaction completed
- Master transaction error
- Slave transaction received
- Slave transaction requested
- Stop condition on bus detected
- Start condition on bus detected

There is a separate interrupt signal for the I²C master and I²C slave modules. While both modules can generate interrupts for multiple conditions, only a single interrupt signal is sent to the interrupt controller.

16.2.3.1 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.

16.2.3.2 I2C Slave Interrupts

The slave module can generate an interrupt when data has been received or requested. This interrupt is enabled by writing a 1 to the DATAIM bit in the **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 DATAIC bit in the **I2C Slave Interrupt Clear (I2CSICR)** register.

In addition, the slave module can generate an interrupt when a start and stop condition is detected. These interrupts are enabled by writing a 1 to the STARTIM and STOPIM bits of the **I2C Slave Interrupt Mask (I2CSIMR)** register and cleared by writing a 1 to the STOPIC and STARTIC bits of the **I2C Slave Interrupt Clear (I2CSICR)** register.

If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I2C Slave Raw Interrupt Status (I2CSRIS)** register.

16.2.4 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.

16.2.5 Command Sequence Flow Charts

This section details the steps required to perform the various I2C transfer types in both master and slave mode.

16.2.5.1 I2C Master Command Sequences

The figures that follow show the command sequences available for the I2C master.

Figure 16-7. Master Single SEND

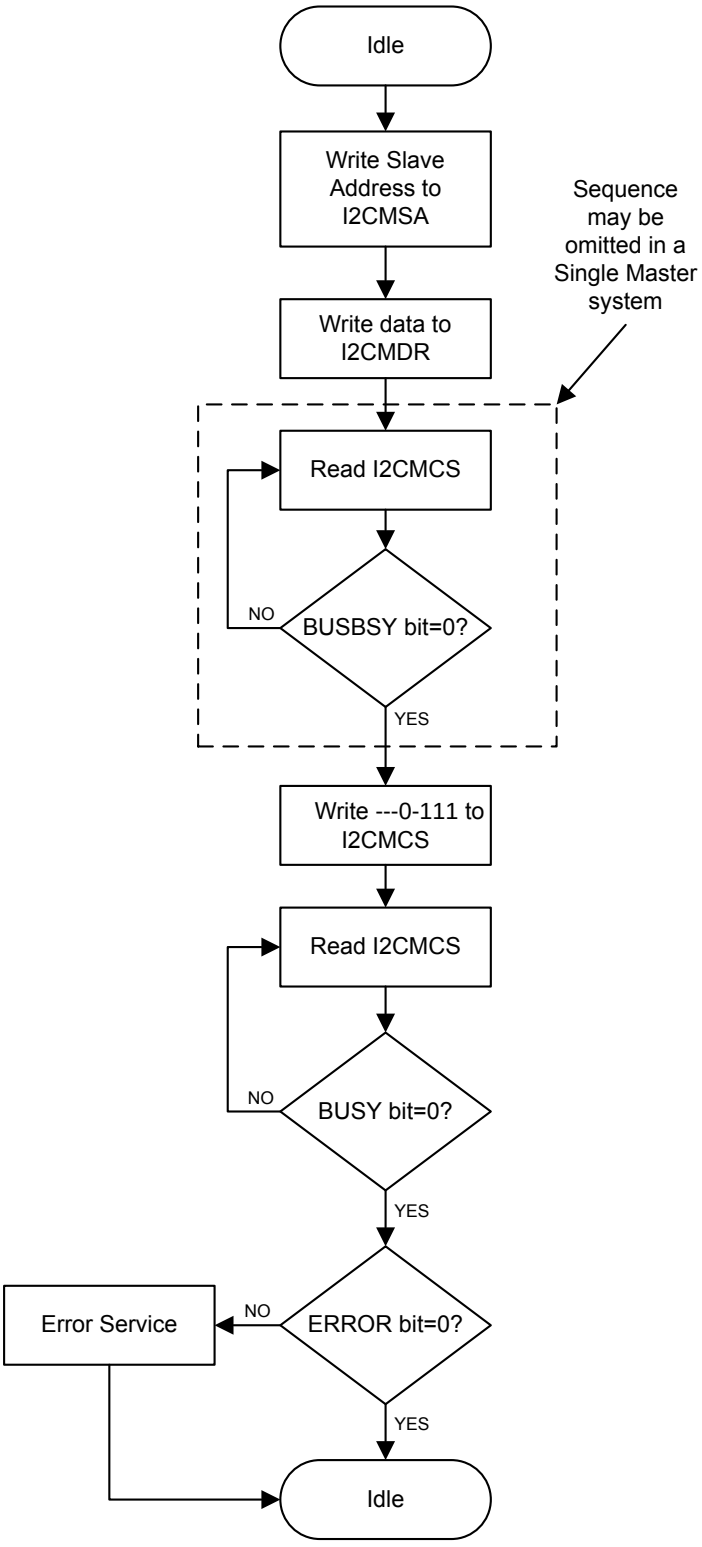


Figure 16-8. Master Single RECEIVE

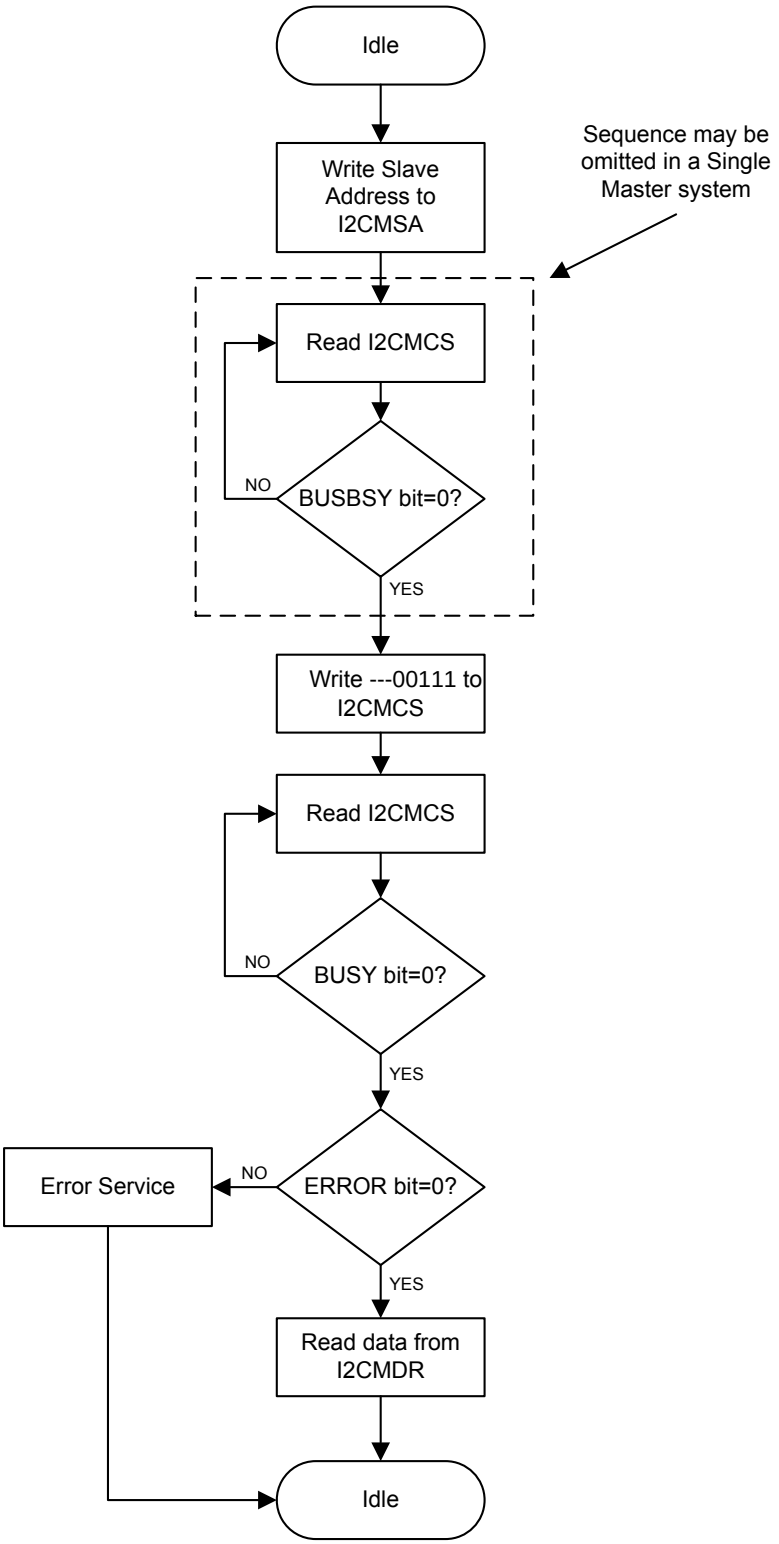


Figure 16-9. Master Burst SEND

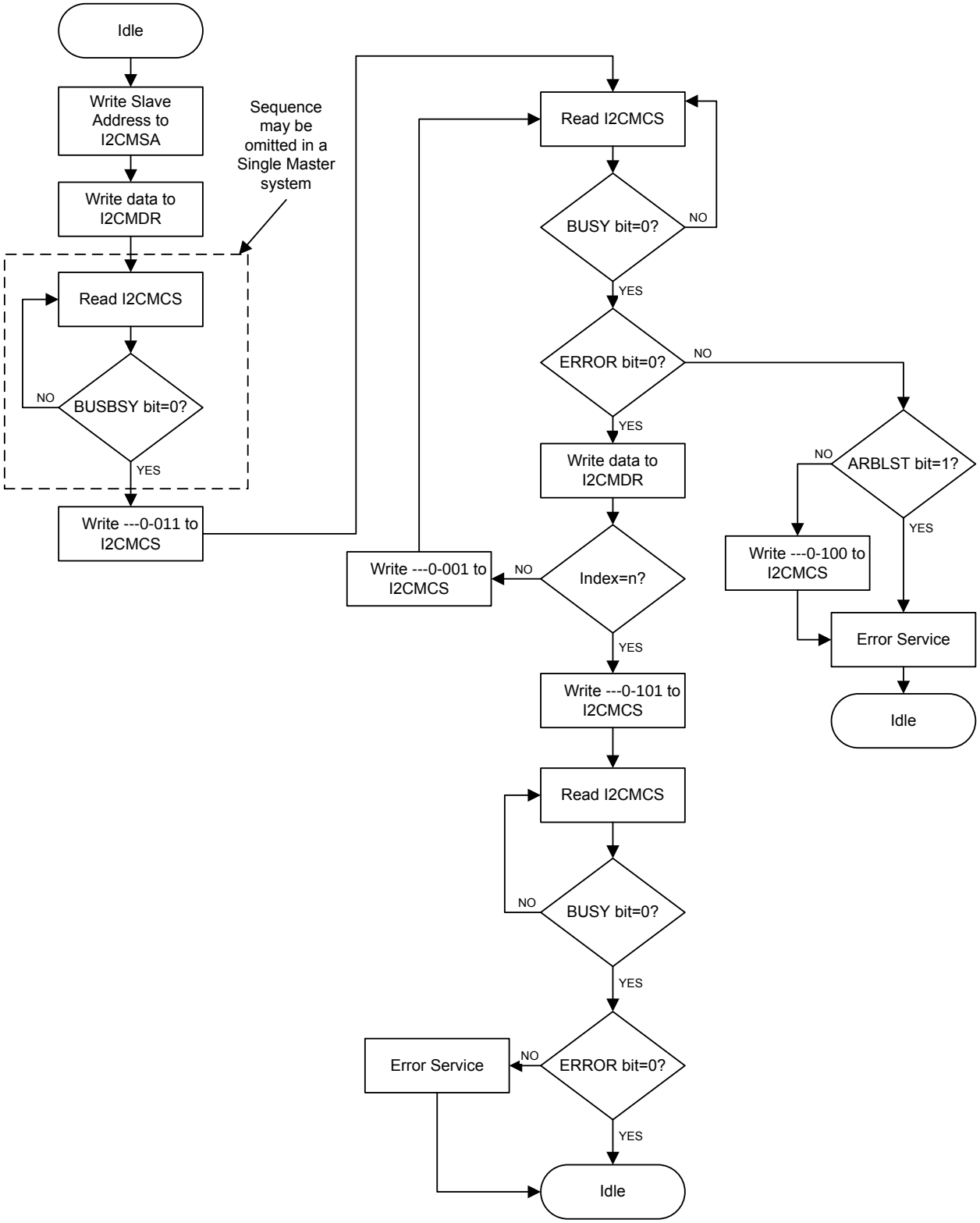


Figure 16-10. Master Burst RECEIVE

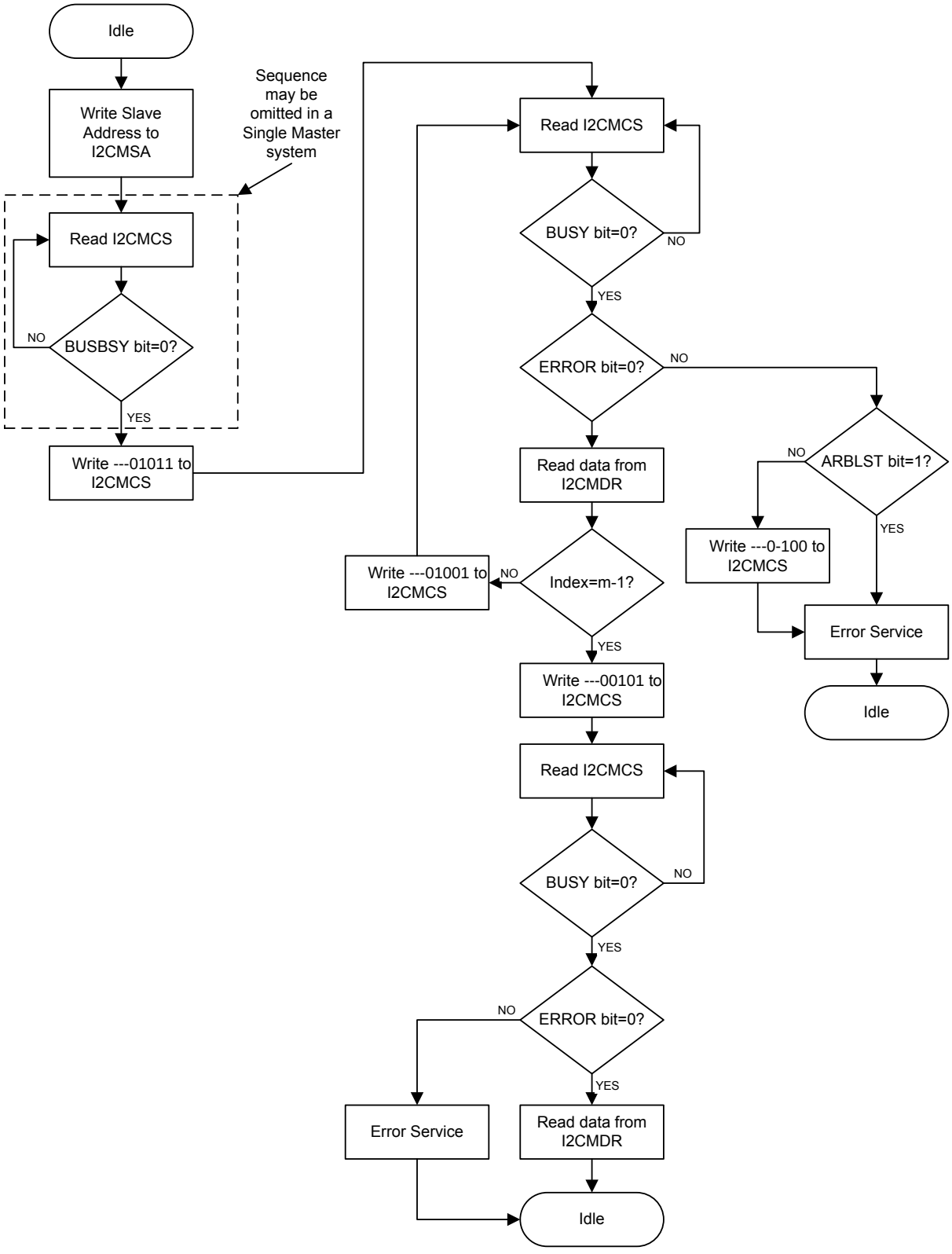


Figure 16-11. Master Burst RECEIVE after Burst SEND

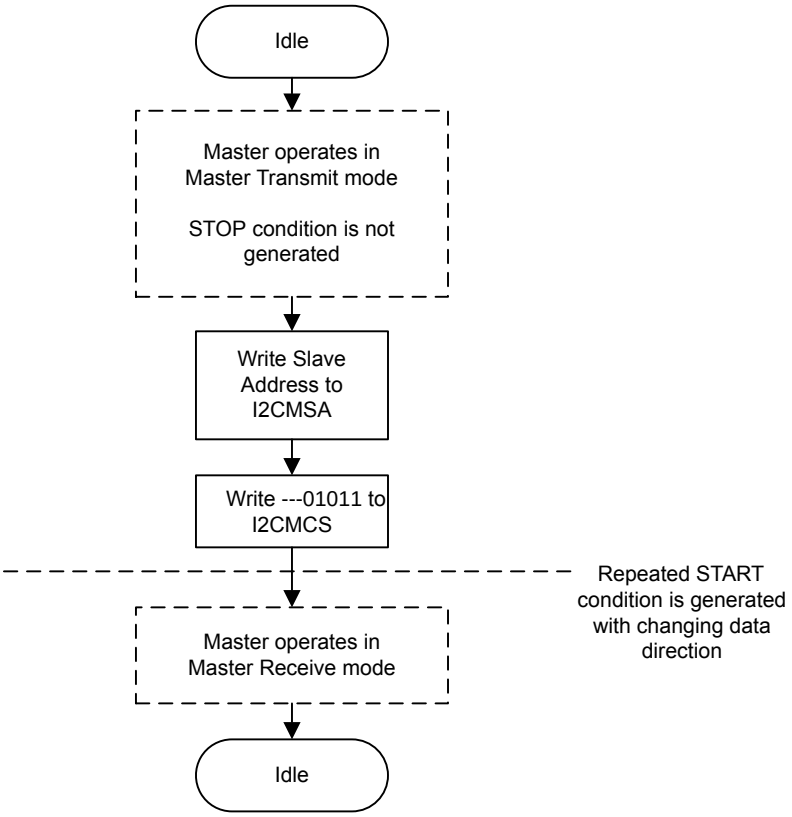
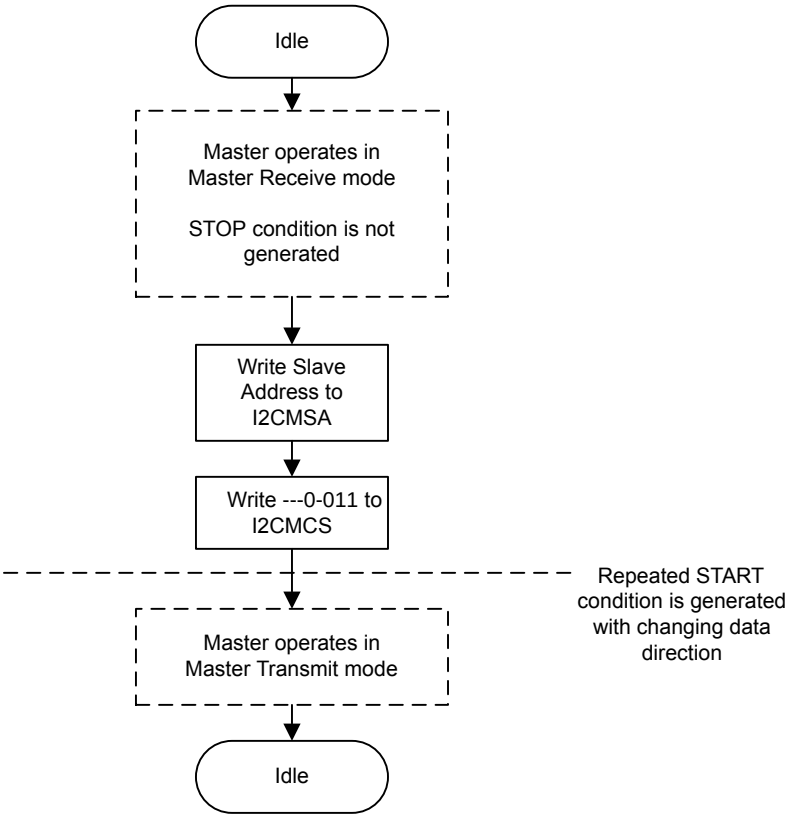


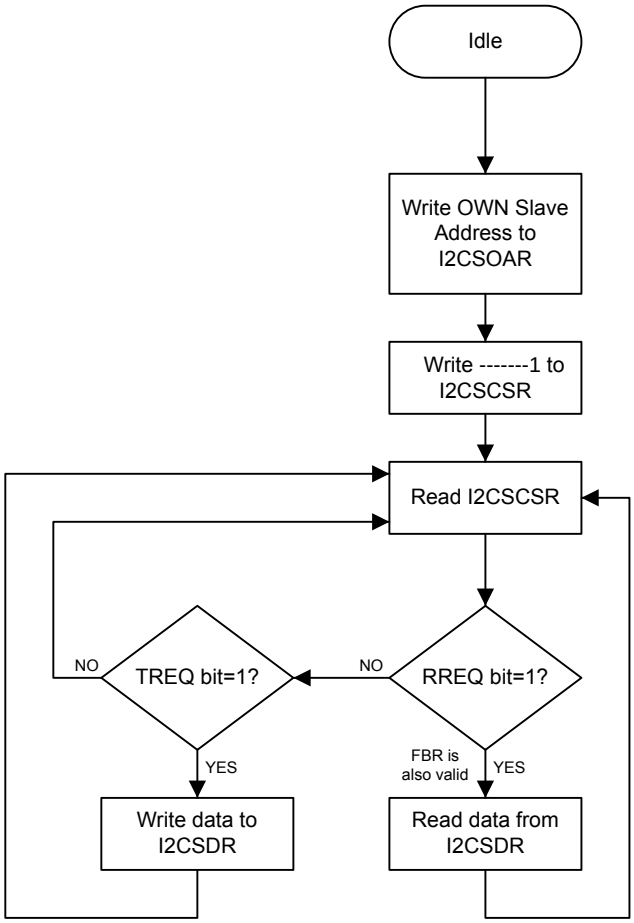
Figure 16-12. Master Burst SEND after Burst RECEIVE



16.2.5.2 I2C Slave Command Sequences

Figure 16-13 on page 491 presents the command sequence available for the I2C slave.

Figure 16-13. Slave Command Sequence



16.3 Initialization and Configuration

The following example shows how to configure the I^2C module to send a single byte as a master. This assumes the system clock is 20 MHz.

1. Enable the I^2C clock by writing a value of 0x0000.1000 to the **RCGC1** register in the System Control module.
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 I^2C Master by writing the **I2CMCR** register with a value of 0x0000.0020.
5. Set the desired SCL clock speed of 100 Kbps by writing the **I2CMTPR** register with the correct value. The value written to the **I2CMTPR** register represents the number of system clock periods in one SCL clock period. The TPR value is determined by the following equation:

```
TPR = (System Clock / (2 * (SCL_LP + SCL_HP) * SCL_CLK)) - 1;
TPR = (20MHz / (2 * (6 + 4) * 100000)) - 1;
TPR = 9
```

Write the **I2CMTPR** register with the value of 0x0000.0009.

6. Specify the slave address of the master and that the next operation will be a Send by writing the **I2CMSA** register with a value of 0x0000.0076. This sets the slave address to 0x3B.
7. Place data (byte) to be sent in the data register by writing the **I2CMDR** register with the desired data.
8. Initiate a single byte send of the data from Master to Slave by writing the **I2CMCS** register with a value of 0x0000.0007 (STOP, START, RUN).
9. Wait until the transmission completes by polling the **I2CMCS** register's BUSBSY bit until it has been cleared.

16.4 Register Map

Table 16-2 on page 492 lists the I2C registers. All addresses given are relative to the I2C base addresses for the master and slave:

- I2C Master 0: 0x4002.0000
- I2C Slave 0: 0x4002.0800
- I2C Master 1: 0x4002.1000
- I2C Slave 1: 0x4002.1800

Table 16-2. Inter-Integrated Circuit (I2C) Interface Register Map

Offset	Name	Type	Reset	Description	See page
I2C Master					
0x000	I2CMSA	R/W	0x0000.0000	I2C Master Slave Address	494
0x004	I2CMCS	R/W	0x0000.0000	I2C Master Control/Status	495
0x008	I2CMDR	R/W	0x0000.0000	I2C Master Data	499
0x00C	I2CMTPR	R/W	0x0000.0001	I2C Master Timer Period	500
0x010	I2CMIMR	R/W	0x0000.0000	I2C Master Interrupt Mask	501
0x014	I2CMRIS	RO	0x0000.0000	I2C Master Raw Interrupt Status	502
0x018	I2CMMIS	RO	0x0000.0000	I2C Master Masked Interrupt Status	503
0x01C	I2CMICR	WO	0x0000.0000	I2C Master Interrupt Clear	504
0x020	I2CMCR	R/W	0x0000.0000	I2C Master Configuration	505
I2C Slave					
0x000	I2CSOAR	R/W	0x0000.0000	I2C Slave Own Address	507
0x004	I2CSCSR	RO	0x0000.0000	I2C Slave Control/Status	508
0x008	I2CSDR	R/W	0x0000.0000	I2C Slave Data	510

Offset	Name	Type	Reset	Description	See page
0x00C	I2CSIMR	R/W	0x0000.0000	I2C Slave Interrupt Mask	511
0x010	I2CSRIS	RO	0x0000.0000	I2C Slave Raw Interrupt Status	512
0x014	I2CSMIS	RO	0x0000.0000	I2C Slave Masked Interrupt Status	513
0x018	I2CSICR	WO	0x0000.0000	I2C Slave Interrupt Clear	514

16.5 Register Descriptions (I^2C Master)

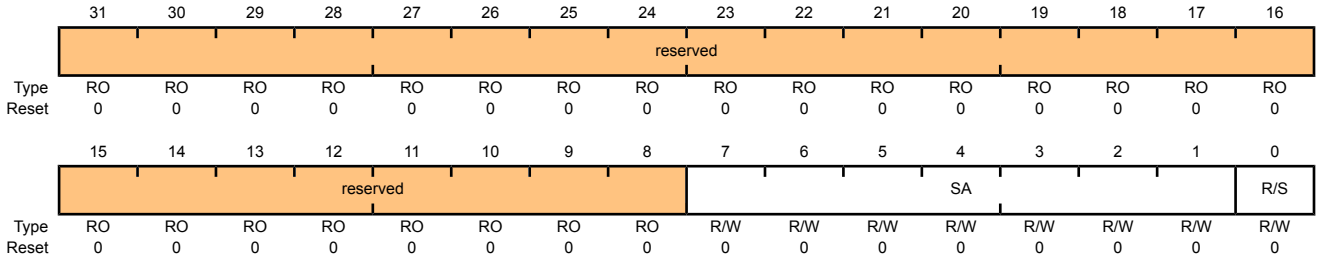
The remainder of this section lists and describes the I^2C master registers, in numerical order by address offset. See also “Register Descriptions (I^2C Slave)” on page 506.

Register 1: I2C Master Slave Address (I2CMSA), offset 0x000

This register consists of eight bits: seven address bits (A6-A0), and a Receive/Send bit, which determines if the next operation is a Receive (High), or Send (Low).

I2C Master Slave Address (I2CMSA)

I2C Master 0 base: 0x4002.0000
I2C Master 1 base: 0x4002.1000
Offset 0x000
Type R/W, reset 0x0000.0000



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). Value Description 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.

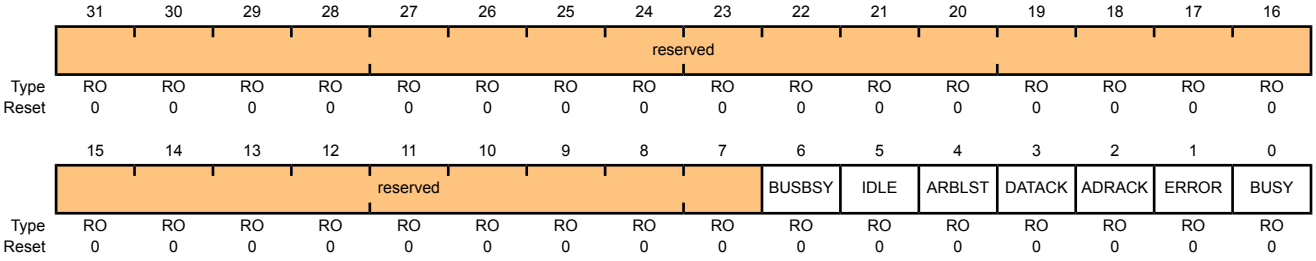
The control register consists of four bits: the RUN, START, STOP, and ACK bits. The START bit causes the generation of the START, or REPEATED START condition.

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.

Reads

I2C Master Control/Status (I2CMCS)

I2C Master 0 base: 0x4002.0000
I2C Master 1 base: 0x4002.1000
Offset 0x004
Type RO, reset 0x0000.0000



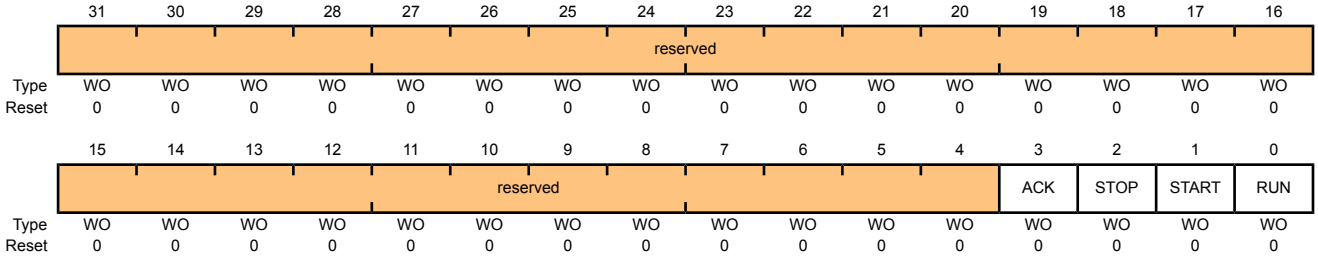
Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	BUSBSY	RO	0	Bus Busy This bit specifies the state 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	I2C 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.

Writes

I2C Master Control/Status (I2CMCS)

I2C Master 0 base: 0x4002.0000
I2C Master 1 base: 0x4002.1000
Offset 0x004
Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	WO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	ACK	WO	0	Data Acknowledge Enable When set, causes received data byte to be acknowledged automatically by the master. See field decoding in Table 16-3 on page 497.
2	STOP	WO	0	Generate STOP When set, causes the generation of the STOP condition. See field decoding in Table 16-3 on page 497.

Bit/Field	Name	Type	Reset	Description
1	START	WO	0	Generate START When set, causes the generation of a START or repeated START condition. See field decoding in Table 16-3 on page 497.
0	RUN	WO	0	I^2C Master Enable When set, allows the master to send or receive data. See field decoding in Table 16-3 on page 497.

Table 16-3. Write Field Decoding for I2CMCS[3:0] Field (Sheet 1 of 3)

Current State	I2CMSA[0]	I2CMCS[3:0]				Description
	R/S	ACK	STOP	START	RUN	
Idle	0	X[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.

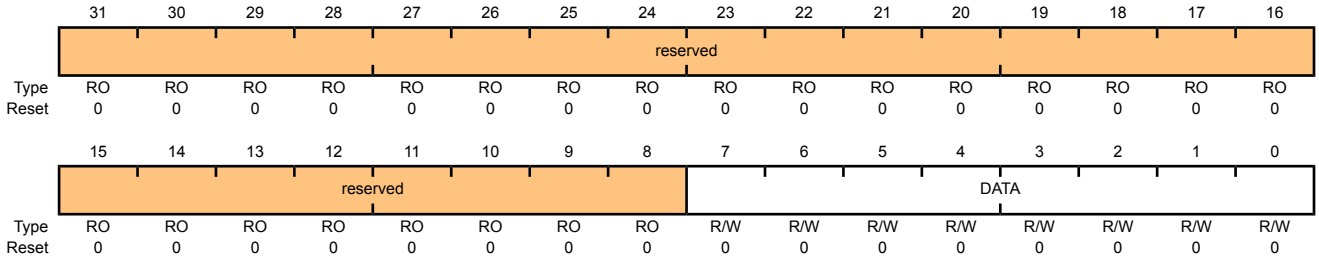
b. In Master Receive mode, a STOP condition should be generated only after a Data Negative Acknowledge executed by the master or an Address Negative Acknowledge executed by the slave.

Register 3: I²C Master Data (I2CMDR), offset 0x008

This register contains the data to be transmitted when in the Master Transmit state, and the data received when in the Master Receive state.

I2C Master Data (I2CMDR)

I2C Master 0 base: 0x4002.0000
I2C Master 1 base: 0x4002.1000
Offset 0x008
Type R/W, reset 0x0000.0000



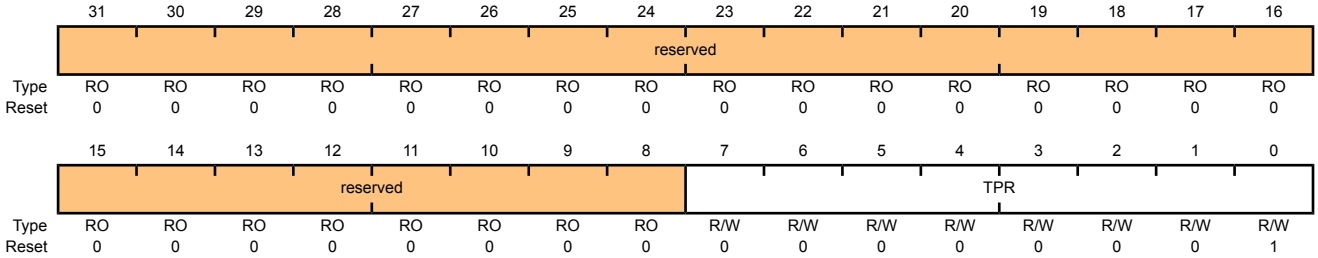
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x00	Data Transferred Data transferred during transaction.

Register 4: I2C Master Timer Period (I2CMTPR), offset 0x00C

This register specifies the period of the SCL clock.

I2C Master Timer Period (I2CMTPR)

I2C Master 0 base: 0x4002.0000
I2C Master 1 base: 0x4002.1000
Offset 0x00C
Type R/W, reset 0x0000.0001



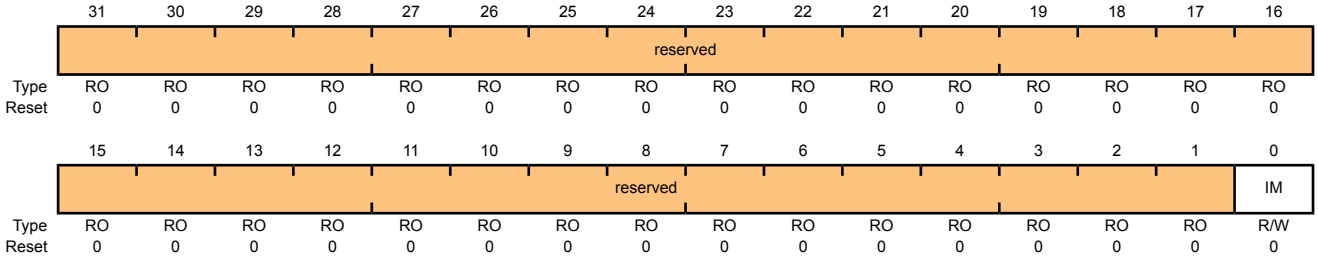
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	TPR	R/W	0x1	SCL Clock Period This field specifies 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 (I2C clock). TPR is the Timer Period register value (range of 1 to 255). SCL_LP is the SCL Low period (fixed at 6). SCL_HP is the SCL High period (fixed at 4).

Register 5: I²C Master Interrupt Mask (I2CMIMR), offset 0x010

This register controls whether a raw interrupt is promoted to a controller interrupt.

I2C Master Interrupt Mask (I2CMIMR)

I2C Master 0 base: 0x4002.0000
I2C Master 1 base: 0x4002.1000
Offset 0x010
Type R/W, reset 0x0000.0000



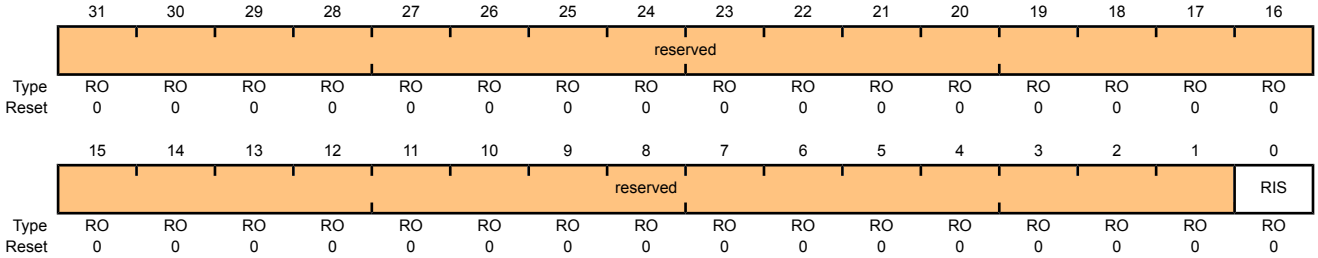
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IM	R/W	0	Interrupt Mask This bit controls whether 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: I²C Master Raw Interrupt Status (I2CMRIS), offset 0x014

This register specifies whether an interrupt is pending.

I2C Master Raw Interrupt Status (I2CMRIS)

I2C Master 0 base: 0x4002.0000
I2C Master 1 base: 0x4002.1000
Offset 0x014
Type RO, reset 0x0000.0000



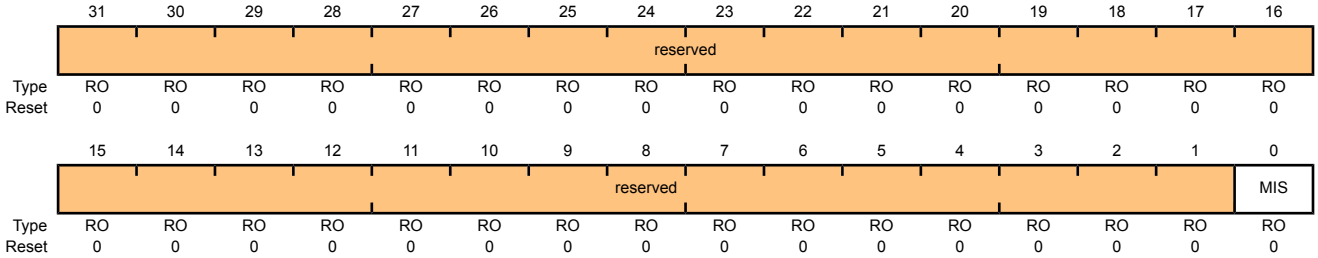
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	RIS	RO	0	Raw Interrupt Status This bit specifies the raw interrupt state (prior to masking) of the I²C master block. If set, an interrupt is pending; otherwise, an interrupt is not pending.

Register 7: I²C Master Masked Interrupt Status (I2CMMIS), offset 0x018

This register specifies whether an interrupt was signaled.

I2C Master Masked Interrupt Status (I2CMMIS)

I2C Master 0 base: 0x4002.0000
I2C Master 1 base: 0x4002.1000
Offset 0x018
Type RO, reset 0x0000.0000



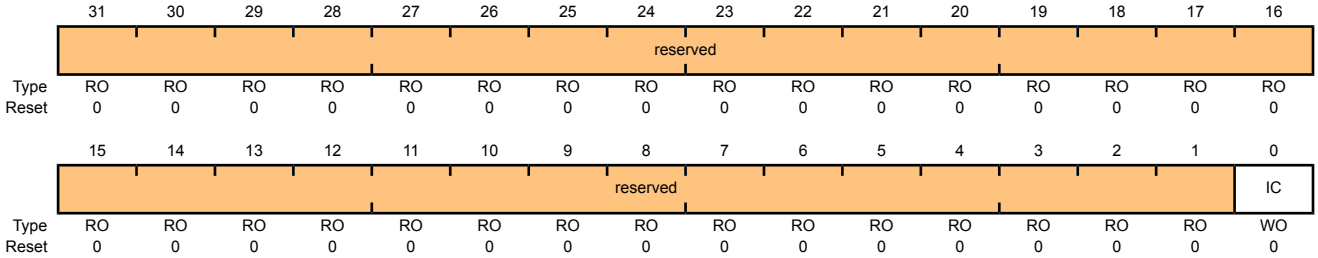
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	MIS	RO	0	Masked Interrupt Status This bit specifies the raw interrupt state (after masking) of the I²C master block. If set, an interrupt was signaled; otherwise, an interrupt has not been generated since the bit was last cleared.

Register 8: I²C Master Interrupt Clear (I2CMICR), offset 0x01C

This register clears the raw interrupt.

I2C Master Interrupt Clear (I2CMICR)

I2C Master 0 base: 0x4002.0000
I2C Master 1 base: 0x4002.1000
Offset 0x01C
Type WO, reset 0x0000.0000



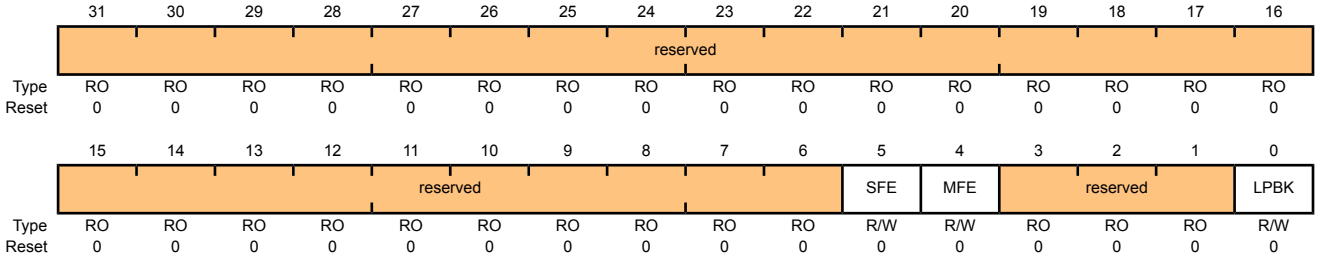
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	IC	WO	0	Interrupt Clear This bit controls the clearing 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: I²C Master Configuration (I2CMCR), offset 0x020

This register configures the mode (Master or Slave) and sets the interface for test mode loopback.

I2C Master Configuration (I2CMCR)

I2C Master 0 base: 0x4002.0000
I2C Master 1 base: 0x4002.1000
Offset 0x020
Type R/W, reset 0x0000.0000



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	I²C 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	I²C 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	I²C Loopback This bit specifies whether the interface is operating normally or in Loopback mode. If set, the device is put in a test mode loopback configuration; otherwise, the device operates normally.

16.6 Register Descriptions (I^2C Slave)

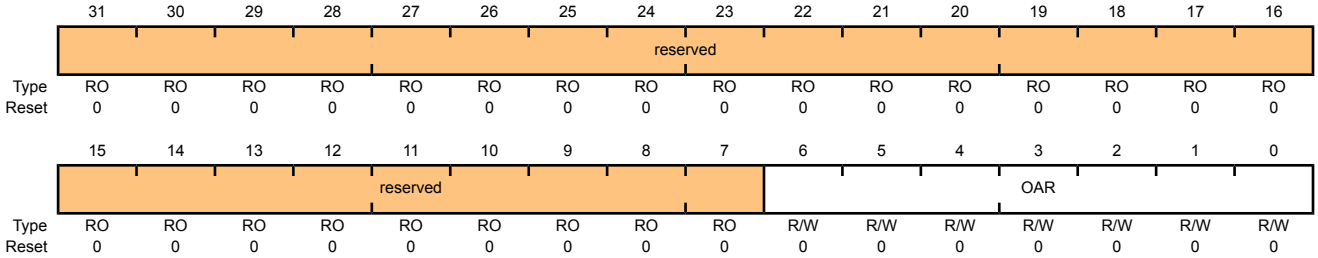
The remainder of this section lists and describes the I^2C slave registers, in numerical order by address offset. See also “Register Descriptions (I^2C Master)” on page 493.

Register 10: I²C Slave Own Address (I2CSOAR), offset 0x000

This register consists of seven address bits that identify the Stellaris® I²C device on the I²C bus.

I2C Slave Own Address (I2CSOAR)

I2C Slave 0 base: 0x4002.0800
I2C Slave 1 base: 0x4002.1800
Offset 0x000
Type R/W, reset 0x0000.0000



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	I²C 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.

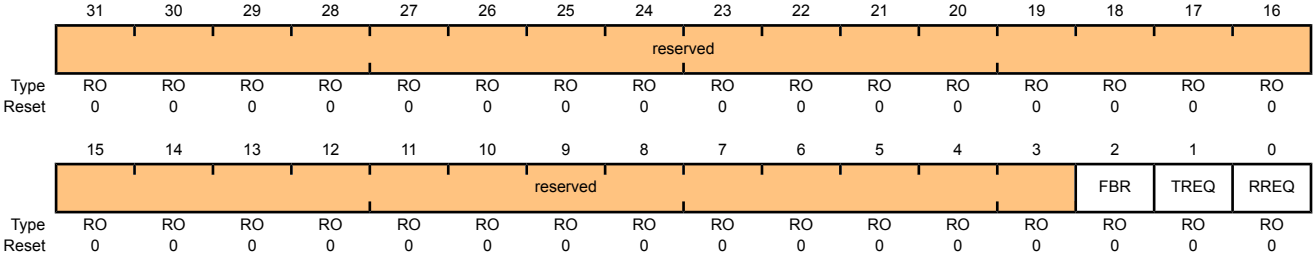
The read-only Status register consists of three bits: the FBR, RREQ, and TREQ bits. The First Byte Received (FBR) bit is set only after the Stellaris® device detects its own slave address and receives the first data byte from the I2C master. The Receive Request (RREQ) bit indicates that the Stellaris® 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.

Reads

I2C Slave Control/Status (I2CSCSR)

I2C Slave 0 base: 0x4002.0800
I2C Slave 1 base: 0x4002.1800
Offset 0x004
Type RO, reset 0x0000.0000



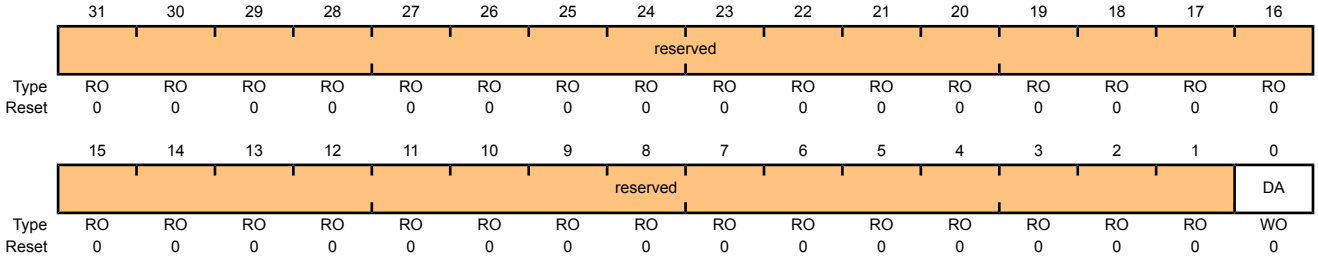
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	FBR	RO	0	First Byte Received Indicates that the first 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.

Bit/Field	Name	Type	Reset	Description
0	RREQ	RO	0	Receive Request This bit specifies the status of the I^{2}C slave with regards to outstanding receive requests. If set, the I^{2}C unit has outstanding receive data from the I^{2}C master and uses clock stretching to delay the master until the data has been read from the **I2CSDR** register. Otherwise, no receive data is outstanding.

Writes

I2C Slave Control/Status (I2CSCSR)

I2C Slave 0 base: 0x4002.0800
I2C Slave 1 base: 0x4002.1800
Offset 0x004
Type WO, reset 0x0000.0000



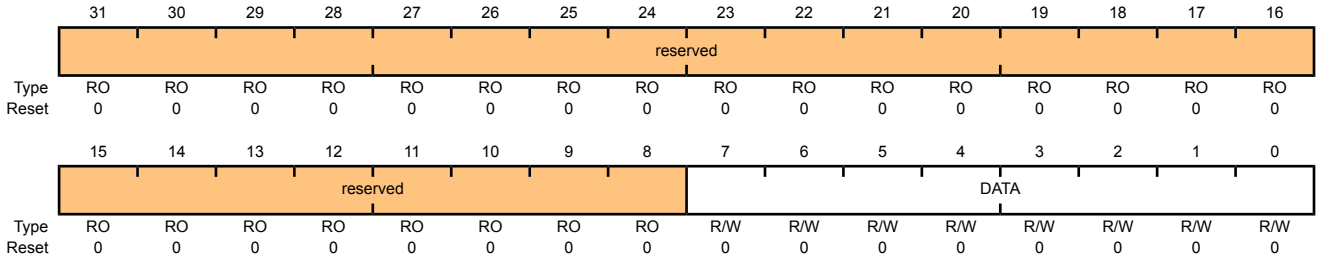
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	DA	WO	0	Device Active Value Description 0 Disables the I^{2}C slave operation. 1 Enables the I^{2}C slave operation.

Register 12: I2C Slave Data (I2CSDR), offset 0x008

This register contains the data to be transmitted when in the Slave Transmit state, and the data received when in the Slave Receive state.

I2C Slave Data (I2CSDR)

I2C Slave 0 base: 0x4002.0800
I2C Slave 1 base: 0x4002.1800
Offset 0x008
Type R/W, reset 0x0000.0000



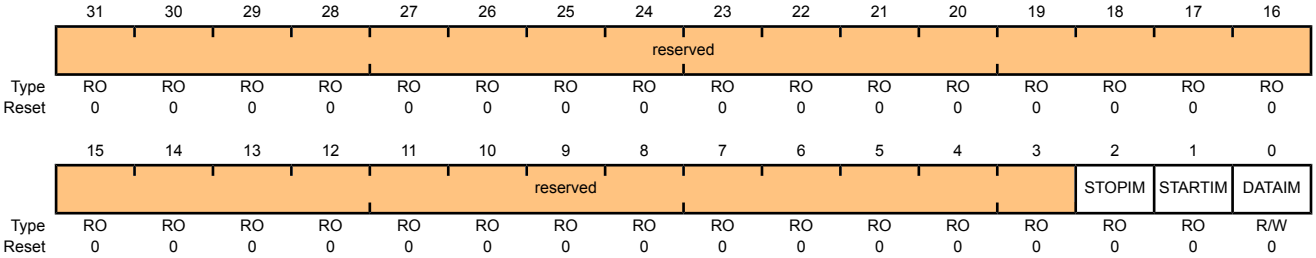
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7:0	DATA	R/W	0x0	Data for Transfer This field contains the data for transfer during a slave receive or transmit operation.

Register 13: I²C Slave Interrupt Mask (I2CSIMR), offset 0x00C

This register controls whether a raw interrupt is promoted to a controller interrupt.

I2C Slave Interrupt Mask (I2CSIMR)

I2C Slave 0 base: 0x4002.0800
I2C Slave 1 base: 0x4002.1800
Offset 0x00C
Type R/W, reset 0x0000.0000



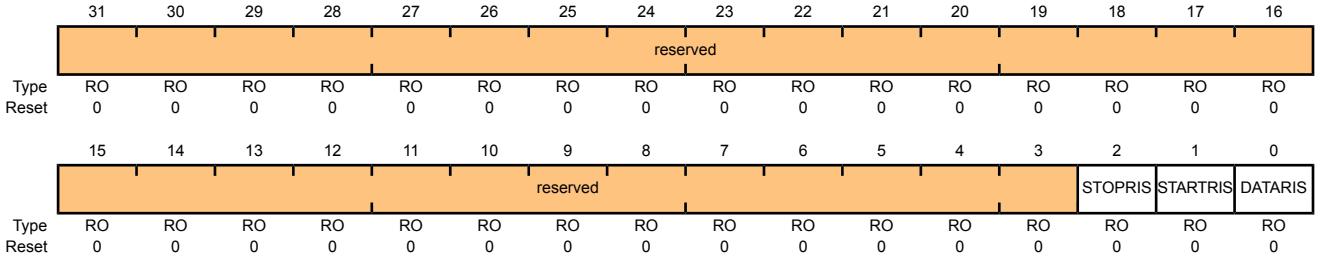
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPIM	RO	0	Stop Condition Interrupt Mask This bit controls whether the raw interrupt for detection of a stop condition on the I²C bus is promoted to a controller interrupt. If set, the interrupt is not masked and the interrupt is promoted; otherwise, the interrupt is masked.
1	STARTIM	RO	0	Start Condition Interrupt Mask This bit controls whether the raw interrupt for detection of a start condition on the I²C bus is promoted to a controller interrupt. If set, the interrupt is not masked and the interrupt is promoted; otherwise, the interrupt is masked.
0	DATAIM	R/W	0	Data Interrupt Mask This bit controls whether the raw interrupt for data received and data requested is promoted to a controller interrupt. If set, the interrupt is not masked and the interrupt is promoted; otherwise, the interrupt is masked.

Register 14: I²C Slave Raw Interrupt Status (I2CSRIS), offset 0x010

This register specifies whether an interrupt is pending.

I2C Slave Raw Interrupt Status (I2CSRIS)

I2C Slave 0 base: 0x4002.0800
I2C Slave 1 base: 0x4002.1800
Offset 0x010
Type RO, reset 0x0000.0000



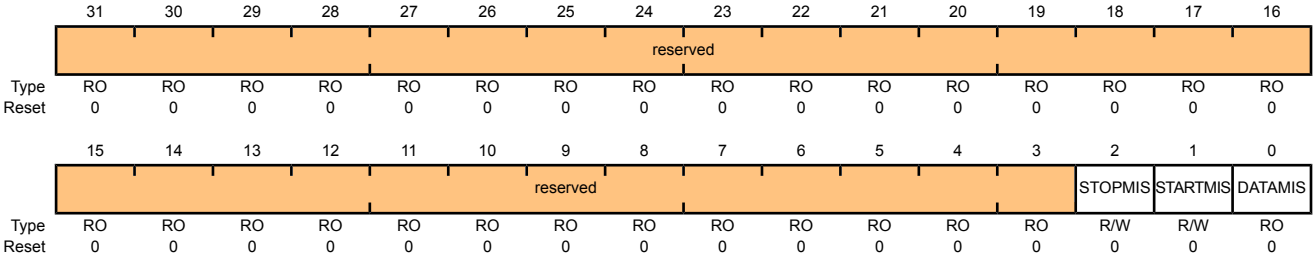
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPRIS	RO	0	Stop Condition Raw Interrupt Status This bit specifies the raw interrupt state for stop condition detect (prior to masking) of the I²C slave block. If set, an interrupt is pending; otherwise, an interrupt is not pending.
1	STARTRIS	RO	0	Start Condition Raw Interrupt Status This bit specifies the raw interrupt state for start condition detect (prior to masking) of the I²C slave block. If set, an interrupt is pending; otherwise, an interrupt is not pending.
0	DATARIS	RO	0	Data Raw Interrupt Status This bit specifies the raw interrupt state for data received and data requested (prior to masking) of the 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
I2C Slave 1 base: 0x4002.1800
Offset 0x014
Type RO, reset 0x0000.0000



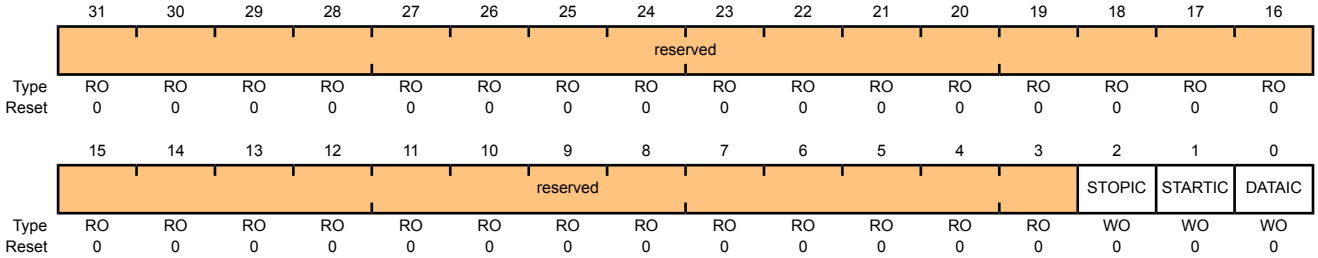
Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPMIS	R/W	0	Stop Condition Masked Interrupt Status This bit specifies the interrupt state for stop condition detect (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.
1	STARTMIS	R/W	0	Start Condition Masked Interrupt Status This bit specifies the interrupt state for start condition detect (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.
0	DATAMIS	RO	0	Data Masked Interrupt Status This bit specifies the interrupt state for data received and data requested (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. A read of this register returns no meaningful data.

I2C Slave Interrupt Clear (I2CSICR)

I2C Slave 0 base: 0x4002.0800
I2C Slave 1 base: 0x4002.1800
Offset 0x018
Type WO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	STOPIC	WO	0	Stop Condition Interrupt Clear This bit controls the clearing of the raw interrupt for stop condition detect. When set, it clears the STOPRIS interrupt bit; otherwise, it has no effect on the STOPRIS bit value.
1	STARTIC	WO	0	Start Condition Interrupt Clear This bit controls the clearing of the raw interrupt for start condition detect. When set, it clears the STARTRIS interrupt bit; otherwise, it has no effect on the STARTRIS bit value.
0	DATAIC	WO	0	Data Interrupt Clear This bit controls the clearing of the raw interrupt for data received and data requested. When set, it clears the DATARIS interrupt bit; otherwise, it has no effect on the DATARIS bit value.

17 Controller Area Network (CAN) Module

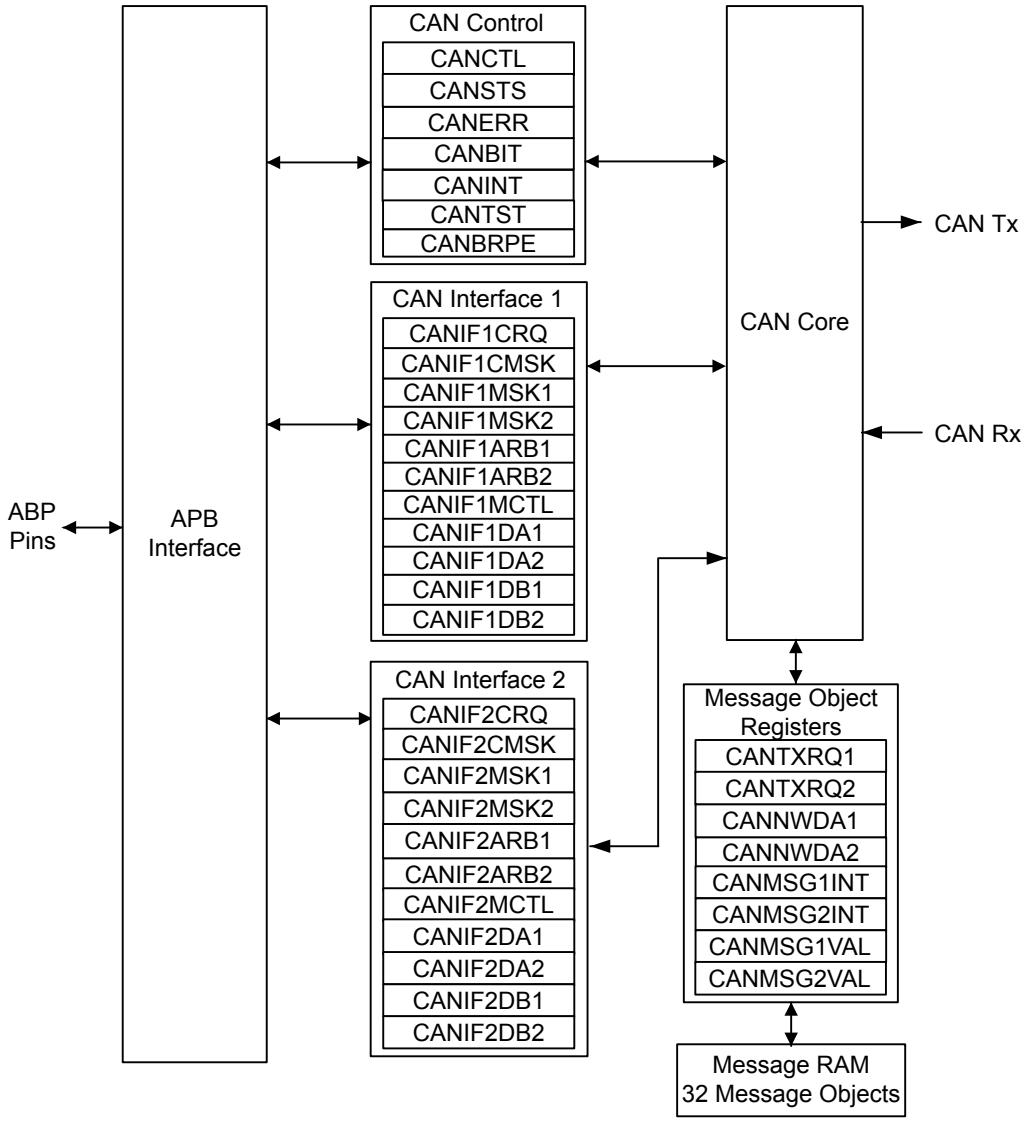
Controller Area Network (CAN) is a multicast, shared serial bus standard for connecting electronic control units (ECUs). CAN was specifically designed to be robust in electromagnetically-noisy environments and can utilize a differential balanced line like RS-485 or a more robust twisted-pair wire. Originally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

Each Stellaris® CAN controller supports the following features:

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

17.1 Block Diagram

Figure 17-1. CAN Controller Block Diagram



17.2 Functional Description

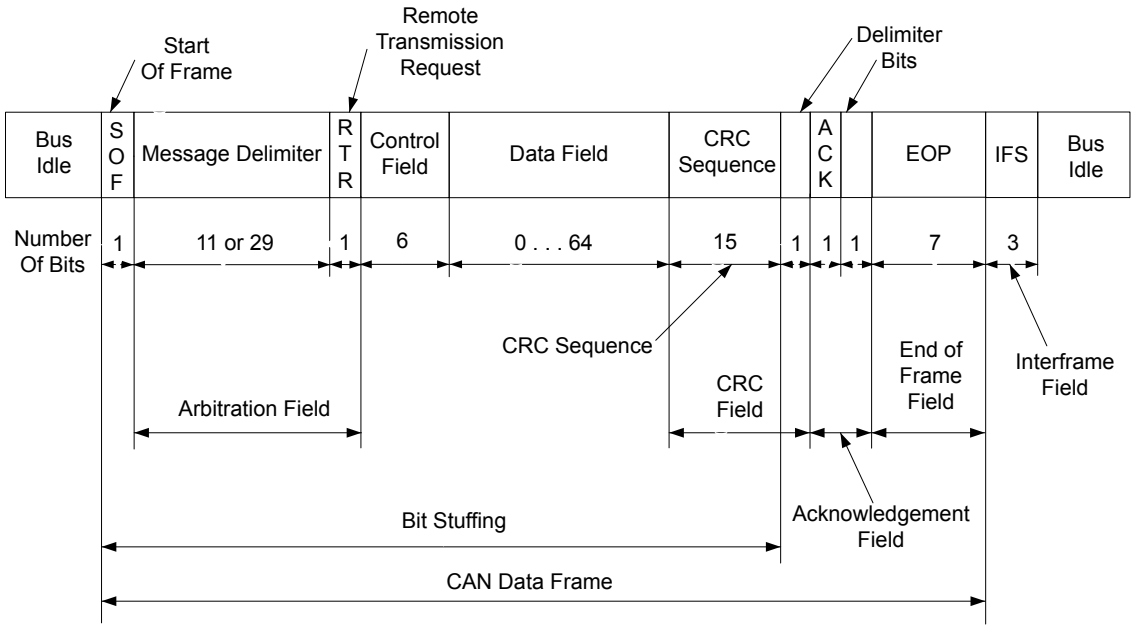
The Stellaris® CAN controller conforms to the CAN protocol version 2.0 (parts A and B). Message transfers that include data, remote, error, and overload frames with an 11-bit identifier (standard) or a 29-bit identifier (extended) are supported. Transfer rates can be programmed up to 1 Mbps.

The CAN module consists of three major parts:

- CAN protocol controller and message handler
- Message memory
- CAN register interface

A data frame contains data for transmission, whereas a remote frame contains no data and is used to request the transmission of a specific message object. The CAN data/remote frame is constructed as shown in Figure 17-2 on page 517.

Figure 17-2. CAN Data/Remote Frame



The protocol controller transfers and receives the serial data from the CAN bus and passes the data on to the message handler. The message handler then loads this information into the appropriate message object based on the current filtering and identifiers in the message object memory. The message handler is also responsible for generating interrupts based on events on the CAN bus.

The message object memory is a set of 32 identical memory blocks that hold the current configuration, status, and actual data for each message object. These are accessed via either of the CAN message object register interfaces.

The message memory is not directly accessible in the Stellaris® memory map, so the Stellaris® CAN controller provides an interface to communicate with the message memory via two CAN interface register sets for communicating with the message objects. As there is no direct access to the message object memory, these two interfaces must be used to read or write to each message object. The two message object interfaces allow parallel access to the CAN controller message objects when multiple objects may have new information that must be processed. In general, one interface is used for transmit data and one for receive data.

17.2.1 Initialization

Software initialization is started by setting the `INIT` bit in the **CAN Control (CANCTL)** register (with software or by a hardware reset) or by going bus-off, which occurs when the transmitter's error counter exceeds a count of 255. While `INIT` is set, all message transfers to and from the CAN bus are stopped and the `CANnTX` signal is held High. Entering the initialization state does not change the configuration of the CAN controller, the message objects, or the error counters. However, some configuration registers are only accessible while in the initialization state.

To initialize the CAN controller, set the **CAN Bit Timing (CANBIT)** register and configure each message object. If a message object is not needed, label it as not valid by clearing the `MSGVAL` bit

in the **CAN IFn Arbitration 2 (CANIFnARB2)** register. Otherwise, the whole message object must be initialized, as the fields of the message object may not have valid information, causing unexpected results. Both the INIT and CCE bits in the **CANCTL** register must be set in order to access the **CANBIT** register and the **CAN Baud Rate Prescaler Extension (CANBRPE)** register to configure the bit timing. To leave the initialization state, the INIT bit must be cleared. Afterwards, the internal Bit Stream Processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (indicating a bus idle condition) before it takes part in bus activities and starts message transfers. Message object initialization does not require the CAN to be in the initialization state and can be done on the fly. However, message objects should all be configured to particular identifiers or set to not valid before message transfer starts. To change the configuration of a message object during normal operation, clear the MSGVAL bit in the **CANIFnARB2** register to indicate that the message object is not valid during the change. When the configuration is completed, set the MSGVAL bit again to indicate that the message object is once again valid.

17.2.2 Operation

There are two sets of CAN Interface Registers (**CANIF1x** and **CANIF2x**), which are used to access the message objects in the Message RAM. The CAN controller coordinates transfers to and from the Message RAM to and from the registers. The two sets are independent and identical and can be used to queue transactions. Generally, one interface is used to transmit data and one is used to receive data.

Once the CAN module is initialized and the INIT bit in the **CANCTL** register is cleared, the CAN module synchronizes itself to the CAN bus and starts the message transfer. As each message is received, it goes through the message handler's filtering process, and if it passes through the filter, is stored in the message object specified by the MNUM bit in the **CAN IFn Command Request (CANIFnCRQ)** register. The whole message (including all arbitration bits, data-length code, and eight data bytes) is stored in the message object. If the Identifier Mask (the MSK bits in the **CAN IFn Mask 1** and **CAN IFn Mask 2 (CANIFnMSKn)** registers) is used, the arbitration bits that are masked to "don't care" may be overwritten in the message object.

The CPU may read or write each message at any time via the CAN Interface Registers. The message handler guarantees data consistency in case of concurrent accesses.

The transmission of message objects is under the control of the software that is managing the CAN hardware. These can be message objects used for one-time data transfers, or permanent message objects used to respond in a more periodic manner. Permanent message objects have all arbitration and control set up, and only the data bytes are updated. At the start of transmission, the appropriate TXRQST bit in the **CAN Transmission Request n (CANTXRQn)** register and the NEWDAT bit in the **CAN New Data n (CANNWDAn)** register are set. If several transmit messages are assigned to the same message object (when the number of message objects is not sufficient), the whole message object has to be configured before the transmission of this message is requested.

The transmission of any number of message objects may be requested at the same time; they are transmitted according to their internal priority, which is based on the message identifier (MNUM) for the message object, with 1 being the highest priority and 32 being the lowest priority. Messages may be updated or set to not valid any time, even when their requested transmission is still pending. The old data is discarded when a message is updated before its pending transmission has started. Depending on the configuration of the message object, the transmission of a message may be requested autonomously by the reception of a remote frame with a matching identifier.

Transmission can be automatically started by the reception of a matching remote frame. To enable this mode, set the RMTEN bit in the **CAN IFn Message Control (CANIFnMCTL)** register. A matching received remote frame causes the TXRQST bit to be set and the message object automatically

transfers its data or generates an interrupt indicating a remote frame was requested. This can be strictly a single message identifier, or it can be a range of values specified in the message object. The CAN mask registers, **CANIFnMSKn**, configure which groups of frames are identified as remote frame requests. The UMASK bit in the **CANIFnMCTL** register enables the MSK bits in the **CANIFnMSKn** register to filter which frames are identified as a remote frame request. The MXTD bit in the **CANIFnMSK2** register should be set if a remote frame request is expected to be triggered by 29-bit extended identifiers.

17.2.3 Transmitting Message Objects

If the internal transmit shift register of the CAN module is ready for loading, and if there is no data transfer occurring between the CAN Interface Registers and message RAM, the valid message object with the highest priority that has a pending transmission request is loaded into the transmit shift register by the message handler and the transmission is started. The message object's NEWDAT bit in the **CANNWDAn** register is cleared. After a successful transmission, and if no new data was written to the message object since the start of the transmission, the TXRQST bit in the **CANTXRQn** register is cleared. If the CAN controller is set up to interrupt upon a successful transmission of a message object, (the TXIE bit in the **CAN IFn Message Control (CANIFnMCTL)** register is set), the INTPND bit in the **CANIFnMCTL** register is set after a successful transmission. If the CAN module has lost the arbitration or if an error occurred during the transmission, the message is re-transmitted as soon as the CAN bus is free again. If, meanwhile, the transmission of a message with higher priority has been requested, the messages are transmitted in the order of their priority.

17.2.4 Configuring a Transmit Message Object

The following steps illustrate how to configure a transmit message object.

1. In the **CAN IFn Command Mask (CANIFnCMASK)** register:
   - Set the WRNRD bit to specify a write to the **CANIFnCMASK** register; specify whether to transfer the IDMASK, DIR, and MXTD of the message object into the **CAN IFn** registers using the MASK bit
   - Specify whether to transfer the ID, DIR, XTD, and MSGVAL of the message object into the interface registers using the ARB bit
   - Specify whether to transfer the control bits into the interface registers using the CONTROL bit
   - Specify whether to clear the INTPND bit in the **CANIFnMCTL** register using the CLRINTPND bit
   - Specify whether to clear the NEWDAT bit in the **CANNWDAn** register using the NEWDAT bit
   - Specify which bits to transfer using the DATAA and DATAB bits
2. In the **CANIFnMSK1** register, use the MSK[15:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[15:0] in this register are used for bits [15:0] of the 29-bit message identifier and are not used for an 11-bit identifier. A value of 0x00 enables all messages to pass through the acceptance filtering. Also note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.

3. In the **CANIFnMSK2** register, use the MSK[12:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[12:0] are used for bits [28:16] of the 29-bit message identifier; whereas MSK[12:2] are used for bits [10:0] of the 11-bit message identifier. Use the MXTD and MDIR bits to specify whether to use XTD and DIR for acceptance filtering. A value of 0x00 enables all messages to pass through the acceptance filtering. Also note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.

4. For a 29-bit identifier, configure ID[15:0] in the **CANIFnARB1** register to are used for bits [15:0] of the message identifier and ID[12:0] in the **CANIFnARB2** register to are used for bits [28:16] of the message identifier. Set the XTD bit to indicate an extended identifier; set the DIR bit to indicate transmit; and set the MSGVAL bit to indicate that the message object is valid.

5. For an 11-bit identifier, disregard the **CANIFnARB1** register and configure ID[12:2] in the **CANIFnARB2** register to are used for bits [10:0] of the message identifier. Clear the XTD bit to indicate a standard identifier; set the DIR bit to indicate transmit; and set the MSGVAL bit to indicate that the message object is valid.

6. In the **CANIFnMCTL** register:

   - Optionally set the UMASK bit to enable the mask (MSK, MXTD, and MDIR specified in the **CANIFnMSK1** and **CANIFnMSK2** registers) for acceptance filtering
   - Optionally set the TXIE bit to enable the INTPND bit to be set after a successful transmission
   - Optionally set the RMTEN bit to enable the TXRQST bit to be set upon the reception of a matching remote frame allowing automatic transmission
   - Set the EOB bit for a single message object;
   - Set the DLC[3:0] field to specify the size of the data frame. Take care during this configuration not to set the NEWDAT, MSGLST, INTPND or TXRQST bits.

7. Load the data to be transmitted into the CAN IFn Data **(CANIFnDA1, CANIFnDA2, CANIFnDB1, CANIFnDB2)** or **(CANIFnDATAA and CANIFnDATAB)** registers. Byte 0 of the CAN data frame is stored in DATA[7:0] in the **CANIFnDA1** register.

8. Program the number of the message object to be transmitted in the MNUM field in the **CAN IFn Command Request (CANIFnCRQ)** register.

9. When everything is properly configured, set the TXRQST bit in the **CANIFnMCTL** register. Once this bit is set, the message object is available to be transmitted, depending on priority and bus availability. Note that setting the RMTEN bit in the **CANIFnMCTL** register can also start message transmission if a matching remote frame has been received.

17.2.5 Updating a Transmit Message Object

The CPU may update the data bytes of a Transmit Message Object any time via the CAN Interface Registers and neither the MSGVAL bit in the **CANIFnARB2** register nor the TXRQST bits in the **CANIFnMCTL** register have to be cleared before the update.

Even if only some of the data bytes are to be updated, all four bytes of the corresponding **CANIFnDAn**/**CANIFnDBn** register have to be valid before the content of that register is transferred to the message object. Either the CPU must write all four bytes into the **CANIFnDAn**/**CANIFnDBn**

register or the message object is transferred to the **CANIFnDAn**/**CANIFnDBn** register before the CPU writes the new data bytes.

In order to only update the data in a message object, the WRNRD, DATAA and DATAB bits in the **CANIFnMSKn** register are set, followed by writing the updated data into **CANIFnDA1**, **CANIFnDA2**, **CANIFnDB1**, and **CANIFnDB2** registers, and then the number of the message object is written to the MNUM field in the **CAN IFn Command Request (CANIFnCRQ)** register. To begin transmission of the new data as soon as possible, set the TXRQST bit in the **CANIFnMSKn** register.

To prevent the clearing of the TXRQST bit in the **CANIFnMCTL** register at the end of a transmission that may already be in progress while the data is updated, the NEWDAT and TXRQST bits have to be set at the same time in the **CANIFnMCTL** register. When these bits are set at the same time, NEWDAT is cleared as soon as the new transmission has started.

17.2.6 Accepting Received Message Objects

When the arbitration and control field (the ID and XTD bits in the **CANIFnARB2** and the RMTEN and DLC[3:0] bits of the **CANIFnMCTL** register) of an incoming message is completely shifted into the CAN controller, the message handling capability of the controller starts scanning the message RAM for a matching valid message object. To scan the message RAM for a matching message object, the controller uses the acceptance filtering programmed through the mask bits in the **CANIFnMSKn** register and enabled using the UMASK bit in the **CANIFnMCTL** register. Each valid message object, starting with object 1, is compared with the incoming message to locate a matching message object in the message RAM. If a match occurs, the scanning is stopped and the message handler proceeds depending on whether it is a data frame or remote frame that was received.

17.2.7 Receiving a Data Frame

The message handler stores the message from the CAN controller receive shift register into the matching message object in the message RAM. The data bytes, all arbitration bits, and the DLC bits are all stored into the corresponding message object. In this manner, the data bytes are connected with the identifier even if arbitration masks are used. The NEWDAT bit of the **CANIFnMCTL** register is set to indicate that new data has been received. The CPU should clear this bit when it reads the message object to indicate to the controller that the message has been received, and the buffer is free to receive more messages. If the CAN controller receives a message and the NEWDAT bit is already set, the MSGLST bit in the **CANIFnMCTL** register is set to indicate that the previous data was lost. If the system requires an interrupt upon successful reception of a frame, the RXIE bit of the **CANIFnMCTL** register should be set. In this case, the INTPND bit of the same register is set, causing the **CANINT** register to point to the message object that just received a message. The TXRQST bit of this message object should be cleared to prevent the transmission of a remote frame.

17.2.8 Receiving a Remote Frame

A remote frame contains no data, but instead specifies which object should be transmitted. When a remote frame is received, three different configurations of the matching message object have to be considered:

Configuration in CANIFnMCTL	Description
■ DIR = 1 (direction = transmit); programmed in the **CANIFnARB2** register ■ RMTEN = 1 (set the TXRQST bit of the **CANIFnMCTL** register at reception of the frame to enable transmission) ■ UMASK = 1 or 0	At the reception of a matching remote frame, the TXRQST bit of this message object is set. The rest of the message object remains unchanged, and the controller automatically transfers the data in the message object as soon as possible.

Configuration in CANIFnMCTL	Description
■ DIR = 1 (direction = transmit); programmed in the **CANIFnARB2** register ■ RMTEN = 0 (do not change the TXRQST bit of the **CANIFnMCTL** register at reception of the frame) ■ UMASK = 0 (ignore mask in the **CANIFnMSKn** register)	At the reception of a matching remote frame, the TXRQST bit of this message object remains unchanged, and the remote frame is ignored. This remote frame is disabled, the data is not transferred and there is no indication that the remote frame ever happened.
■ DIR = 1 (direction = transmit); programmed in the **CANIFnARB2** register ■ RMTEN = 0 (do not change the TXRQST bit of the **CANIFnMCTL** register at reception of the frame) ■ UMASK = 1 (use mask (MSK, MXTD, and MDIR in the **CANIFnMSKn** register) for acceptance filtering)	At the reception of a matching remote frame, the TXRQST bit of this message object is cleared. The arbitration and control field (ID + XTD + RMTEN + DLC) from the shift register is stored into the message object in the message RAM and the NEWDAT bit of this message object is set. The data field of the message object remains unchanged; the remote frame is treated similar to a received data frame. This is useful for a remote data request from another CAN device for which the Stellaris® controller does not have readily available data. The software must fill the data and answer the frame manually.

17.2.9 Receive/Transmit Priority

The receive/transmit priority for the message objects is controlled by the message number. Message object 1 has the highest priority, while message object 32 has the lowest priority. If more than one transmission request is pending, the message objects are transmitted in order based on the message object with the lowest message number. This should not be confused with the message identifier as that priority is enforced by the CAN bus. This means that if message object 1 and message object 2 both have valid messages that need to be transmitted, message object 1 will always be transmitted first regardless of the message identifier in the message object itself.

17.2.10 Configuring a Receive Message Object

The following steps illustrate how to configure a receive message object.

1. Program the **CAN IFn Command Mask (CANIFnCMASK)** register as described in the "Configuring a Transmit Message Object" on page 519 section, except that the WRNRD bit is set to specify a write to the message RAM.

2. Program the **CANIFnMSK1**and **CANIFnMSK2** registers as described in the "Configuring a Transmit Message Object" on page 519 section to configure which bits are used for acceptance filtering. Note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.

3. In the **CANIFnMSK2** register, use the MSK[12:0] bits to specify which of the bits in the 29-bit or 11-bit message identifier are used for acceptance filtering. Note that MSK[12:0] are used for bits [28:16] of the 29-bit message identifier; whereas MSK[12:2] are used for bits [10:0] of the 11-bit message identifier. Use the MXTD and MDIR bits to specify whether to use XTD and DIR for acceptance filtering. A value of 0x00 enables all messages to pass through the acceptance filtering. Also note that in order for these bits to be used for acceptance filtering, they must be enabled by setting the UMASK bit in the **CANIFnMCTL** register.

4. Program the **CANIFnARB1** and **CANIFnARB2** registers as described in the "Configuring a Transmit Message Object" on page 519 section to program XTD and ID bits for the message identifier to be received; set the MSGVAL bit to indicate a valid message; and clear the DIR bit to specify receive.

5. In the **CANIFnMCTL** register:

   - Optionally set the UMASK bit to enable the mask (MSK, MXTD, and MDIR specified in the **CANIFnMSK1** and **CANIFnMSK2** registers) for acceptance filtering
   - Optionally set the RXIE bit to enable the INTPND bit to be set after a successful reception
   - Clear the RMTEN bit to leave the TXRQST bit unchanged
   - Set the EOB bit for a single message object
   - Set the DLC[3:0] field to specify the size of the data frame

   Take care during this configuration not to set the NEWDAT, MSGLST, INTPND or TXRQST bits.

6. Program the number of the message object to be received in the MNUM field in the **CAN IFn Command Request (CANIFnCRQ)** register. Reception of the message object begins as soon as a matching frame is available on the CAN bus.

When the message handler stores a data frame in the message object, it stores the received Data Length Code and eight data bytes in the **CANIFnDA1**, **CANIFnDA2**, **CANIFnDB1**, and **CANIFnDB2** register. Byte 0 of the CAN data frame is stored in DATA[7:0] in the **CANIFnDA1** register. If the Data Length Code is less than 8, the remaining bytes of the message object are overwritten by unspecified values.

The CAN mask registers can be used to allow groups of data frames to be received by a message object. The CAN mask registers, **CANIFnMSKn**, configure which groups of frames are received by a message object. The UMASK bit in the **CANIFnMCTL** register enables the MSK bits in the **CANIFnMSKn** register to filter which frames are received. The MXTD bit in the **CANIFnMSK2** register should be set if only 29-bit extended identifiers are expected by this message object.

17.2.11 Handling of Received Message Objects

The CPU may read a received message any time via the CAN Interface registers because the data consistency is guaranteed by the message handler state machine.

Typically, the CPU first writes 0x007F to the **CANIFnCMSK** register and then writes the number of the message object to the **CANIFnCRQ** register. That combination transfers the whole received message from the message RAM into the Message Buffer registers (**CANIFnMSKn**, **CANIFnARBn**, and **CANIFnMCTL**). Additionally, the NEWDAT and INTPND bits are cleared in the message RAM, acknowledging that the message has been read and clearing the pending interrupt generated by this message object.

If the message object uses masks for acceptance filtering, the **CANIFnARBn** registers show the full, unmasked ID for the received message.

The NEWDAT bit in the **CANIFnMCTL** register shows whether a new message has been received since the last time this message object was read. The MSGLST bit in the **CANIFnMCTL** register shows whether more than one message has been received since the last time this message object was read. MSGLST is not automatically cleared, and should be cleared by software after reading its status.

Using a remote frame, the CPU may request new data from another CAN node on the CAN bus. Setting the TXRQST bit of a receive object causes the transmission of a remote frame with the receive object's identifier. This remote frame triggers the other CAN node to start the transmission of the matching data frame. If the matching data frame is received before the remote frame could be

transmitted, the TXRQST bit is automatically reset. This prevents the possible loss of data when the other device on the CAN bus has already transmitted the data slightly earlier than expected.

17.2.11.1 Configuration of a FIFO Buffer

With the exception of the EOB bit in the **CANIFnMCTL** register, the configuration of receive message objects belonging to a FIFO buffer is the same as the configuration of a single receive message object (see “Configuring a Receive Message Object” on page 522). To concatenate two or more message objects into a FIFO buffer, the identifiers and masks (if used) of these message objects have to be programmed to matching values. Due to the implicit priority of the message objects, the message object with the lowest message object number is the first message object in a FIFO buffer. The EOB bit of all message objects of a FIFO buffer except the last one must be cleared. The EOB bit of the last message object of a FIFO buffer is set, indicating it is the last entry in the buffer.

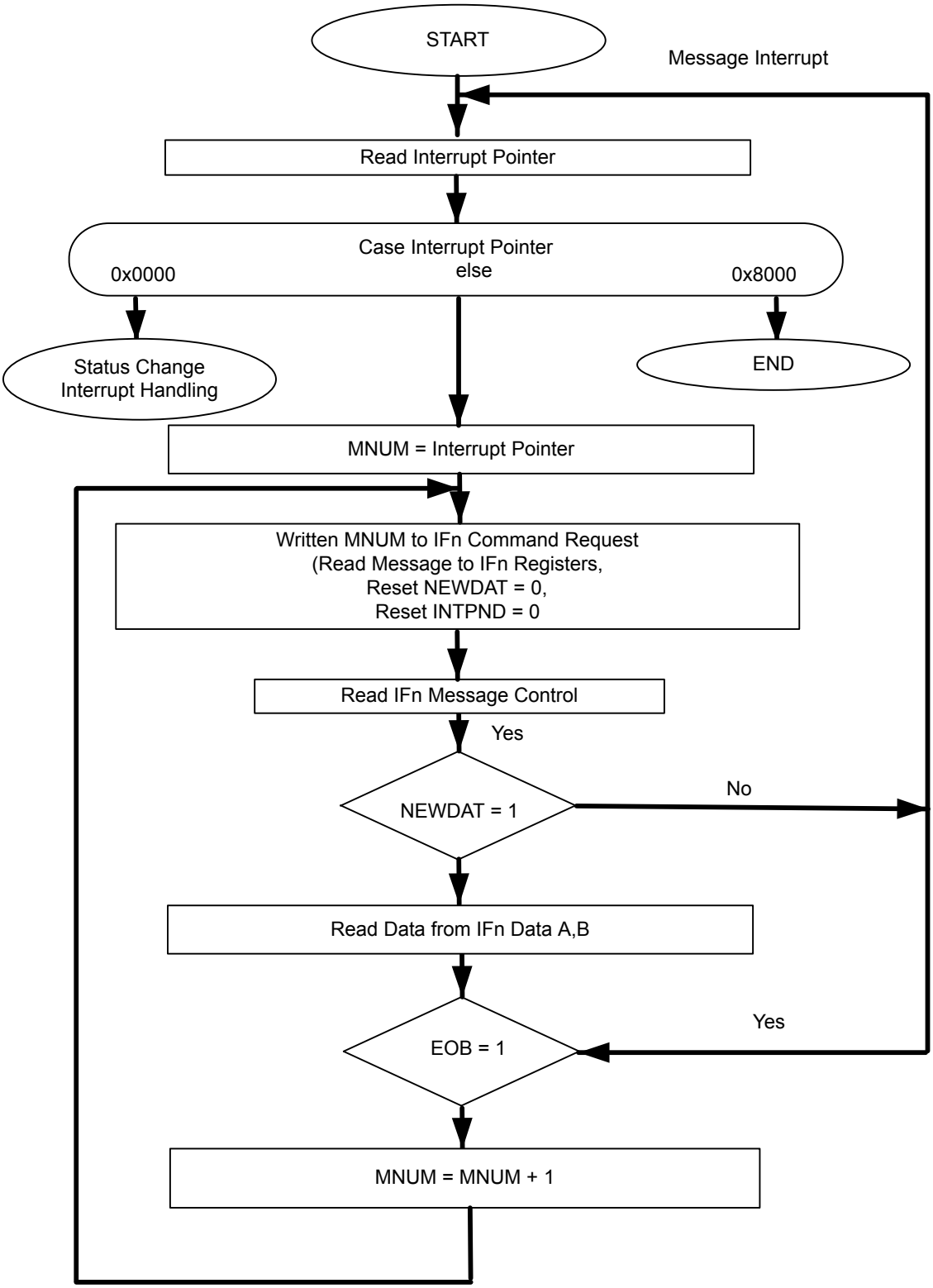
17.2.11.2 Reception of Messages with FIFO Buffers

Received messages with identifiers matching to a FIFO buffer are stored starting with the message object with the lowest message number. When a message is stored into a message object of a FIFO buffer, the NEWDAT of the **CANIFnMCTL** register bit of this message object is set. By setting NEWDAT while EOB is clear, the message object is locked and cannot be written to by the message handler until the CPU has cleared the NEWDAT bit. Messages are stored into a FIFO buffer until the last message object of this FIFO buffer is reached. If none of the preceding message objects has been released by clearing the NEWDAT bit, all further messages for this FIFO buffer will be written into the last message object of the FIFO buffer and therefore overwrite previous messages.

17.2.11.3 Reading from a FIFO Buffer

When the CPU transfers the contents of a message object from a FIFO buffer by writing its number to the **CANIFnCRQ**, the TXRQST and CLRINTPND bits in the **CANIFnCMSK** register should be set such that the NEWDAT and INTPEND bits in the **CANIFnMCTL** register are cleared after the read. The values of these bits in the **CANIFnMCTL** register always reflect the status of the message object before the bits are cleared. To assure the correct function of a FIFO buffer, the CPU should read out the message objects starting with the message object with the lowest message number. Figure 17-3 on page 525 shows how a set of message objects which are concatenated to a FIFO Buffer can be handled by the CPU.

Figure 17-3. Message Objects in a FIFO Buffer



17.2.12 Handling of Interrupts

If several interrupts are pending, the **CAN Interrupt (CANINT)** register points to the pending interrupt with the highest priority, disregarding their chronological order. The status interrupt has the highest priority. Among the message interrupts, the message object's interrupt with the lowest message number has the highest priority. A message interrupt is cleared by clearing the message object's INTPND bit in the **CANIFnMCTL** register or by reading the **CAN Status (CANSTS)** register. The status Interrupt is cleared by reading the **CANSTS** register.

The interrupt identifier INTID in the **CANINT** register indicates the cause of the interrupt. When no interrupt is pending, the register reads as 0x0000. If the value of the INTID field is different from 0, then there is an interrupt pending. If the IE bit is set in the **CANCTL** register, the interrupt line to the CPU is active. The interrupt line remains active until the INTID field is 0, meaning that all interrupt sources have been cleared (the cause of the interrupt is reset), or until IE is cleared, which disables interrupts from the CAN controller.

The INTID field of the **CANINT** register points to the pending message interrupt with the highest interrupt priority. The SIE bit in the **CANCTL** register controls whether a change of the RXOK, TXOK, and LEC bits in the **CANSTS** can cause an interrupt. The EIE bit in the **CANCTL**register controls whether a change of the BOFF and EWARN bits in the **CANSTS** can cause an interrupt. The IE bit in the **CANCTL** controls whether any interrupt from the CAN controller actually generates an interrupt to the microcontroller's interrupt controller. The **CANINT** register is updated even when the IE bit in the **CANCTL** register is clear, but the interrupt will not be indicated to the CPU.

A value of 0x8000 in the **CANINT** register indicates that an interrupt is pending because the CAN module has updated, but not necessarily changed, the **CANSTS** , indicating that either an error or status interrupt has been generated. A write access to the **CANSTS** register can clear the RXOK, TXOK, and LEC bits in that same register; however, the only way to clear the source of a status interrupt is to read the **CANSTS** register.

There are two ways to determine the source of an interrupt during interrupt handling. The first is to read the INTID bit in the **CANINT** register to determine the highest priority interrupt that is pending, and the second is to read the **CAN Message Interrupt Pending (CANMSGnINT)** register to see all of the message objects that have pending interrupts.

An interrupt service routine reading the message that is the source of the interrupt may read the message and clear the message object's INTPND bit at the same time by setting the CLRINTPND bit in the **CANIFnCMSK** register. Once the INTPND bit has been cleared, the **CANINT** register contains the message number for the next message object with a pending interrupt.

17.2.13 Test Mode

A Test Mode is provided, which allows various diagnostics to be performed. Test Mode is entered by setting the TEST bit **CANCTL** register. Once in Test Mode, the TX[1:0], LBACK, SILENT and BASIC bits in the **CAN Test (CANTST)** register can be used to put the CAN controller into the various diagnostic modes. The RX bit in the **CANTST** register allows monitoring of the CANnRX signal. All **CANTST** register functions are disabled when the TEST bit is cleared.

17.2.13.1 Silent Mode

Silent Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames). The CAN Controller is put in Silent Mode setting the SILENT bit in the **CANTST** register. In Silent Mode, the CAN controller is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the CAN Controller is required to send a dominant bit (ACK bit, overload flag,

or active error flag), the bit is rerouted internally so that the CAN Controller monitors this dominant bit, although the CAN bus remains in recessive state.

17.2.13.2 Loopback Mode

Loopback mode is useful for self-test functions. In Loopback Mode, the CAN Controller internally routes the CANnTX signal on to the CANnRX signal and treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into the message buffer. The CAN Controller is put in Loopback Mode by setting the LBACK bit in the **CANTST** register. To be independent from external stimulation, the CAN Controller ignores acknowledge errors (a recessive bit sampled in the acknowledge slot of a data/remote frame) in Loopback Mode. The actual value of the CANnRX signal is disregarded by the CAN Controller. The transmitted messages can be monitored on the CANnTX signal.

17.2.13.3 Loopback Combined with Silent Mode

Loopback Mode and Silent Mode can be combined to allow the CAN Controller to be tested without affecting a running CAN system connected to the CANnTX and CANnRX signals. In this mode, the CANnRX signal is disconnected from the CAN Controller and the CANnTX signal is held recessive. This mode is enabled by setting both the LBACK and SILENT bits in the **CANTST** register.

17.2.13.4 Basic Mode

Basic Mode allows the CAN Controller to be operated without the Message RAM. In Basic Mode, The CANIF1 registers are used as the transmit buffer. The transmission of the contents of the IF1 registers is requested by setting the BUSY bit of the **CANIF1CRQ** register. The CANIF1 registers are locked while the BUSY bit is set. The BUSY bit indicates that a transmission is pending. As soon the CAN bus is idle, the CANIF1 registers are loaded into the shift register of the CAN Controller and transmission is started. When the transmission has completed, the BUSY bit is cleared and the locked CANIF1 registers are released. A pending transmission can be aborted at any time by clearing the BUSY bit in the **CANIF1CRQ** register while the CANIF1 registers are locked. If the CPU has cleared the BUSY bit, a possible retransmission in case of lost arbitration or an error is disabled.

The CANIF2 Registers are used as a receive buffer. After the reception of a message, the contents of the shift register is stored into the CANIF2 registers, without any acceptance filtering. Additionally, the actual contents of the shift register can be monitored during the message transfer. Each time a read message object is initiated by setting the BUSY bit of the **CANIF2CRQ** register, the contents of the shift register are stored into the CANIF2 registers.

In Basic Mode, all message-object-related control and status bits and of the control bits of the **CANIFnCMSK** registers are not evaluated. The message number of the **CANIFnCRQ** registers is also not evaluated. In the **CANIF2MCTL** register, the NEWDAT and MSGLST bits retain their function, the DLC[3:0] field shows the received DLC, the other control bits are cleared.

Basic Mode is enabled by setting the BASIC bit in the **CANTST** register.

17.2.13.5 Transmit Control

Software can directly override control of the CANnTX signal in four different ways.

- CANnTX is controlled by the CAN Controller
- The sample point is driven on the CANnTX signal to monitor the bit timing
- CANnTX drives a low value
- CANnTX drives a high value

The last two functions, combined with the readable CAN receive pin CANnRX, can be used to check the physical layer of the CAN bus.

The Transmit Control function is enabled by programming the TX[1:0] field in the **CANTST** register. The three test functions for the CANnTX signal interfere with all CAN protocol functions. TX[1:0] must be cleared when CAN message transfer or Loopback Mode, Silent Mode, or Basic Mode are selected.

17.2.14 Bit Timing Configuration Error Considerations

Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly. In many cases, the CAN bit synchronization amends a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. In the case of arbitration, however, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive. The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and of the CAN nodes' interaction on the CAN bus.

17.2.15 Bit Time and Bit Rate

The CAN system supports bit rates in the range of lower than 1 Kbps up to 1000 Kbps. Each member of the CAN network has its own clock generator. The timing parameter of the bit time can be configured individually for each CAN node, creating a common bit rate even though the CAN nodes' oscillator periods may be different.

Because of small variations in frequency caused by changes in temperature or voltage and by deteriorating components, these oscillators are not absolutely stable. As long as the variations remain inside a specific oscillator's tolerance range, the CAN nodes are able to compensate for the different bit rates by periodically resynchronizing to the bit stream.

According to the CAN specification, the bit time is divided into four segments (see Figure 17-4 on page 529): the Synchronization Segment, the Propagation Time Segment, the Phase Buffer Segment 1, and the Phase Buffer Segment 2. Each segment consists of a specific, programmable number of time quanta (see Table 17-1 on page 529). The length of the time quantum (t_q), which is the basic time unit of the bit time, is defined by the CAN controller's system clock (fsys) and the Baud Rate Prescaler (BRP):

t_q = BRP / fsys

The CAN module's system clock fsys is the frequency of its CAN module clock input.

The Synchronization Segment Sync_Seg is that part of the bit time where edges of the CAN bus level are expected to occur; the distance between an edge that occurs outside of Sync_Seg and the Sync_Seg is called the *phase error* of that edge.

The Propagation Time Segment Prop_Seg is intended to compensate for the physical delay times within the CAN network.

The Phase Buffer Segments Phase_Seg1 and Phase_Seg2 surround the Sample Point.

The (Re-)Synchronization Jump Width (SJW) defines how far a resynchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

A given bit rate may be met by different bit-time configurations, but for the proper function of the CAN network, the physical delay times and the oscillator's tolerance range have to be considered.

Figure 17-4. CAN Bit Time

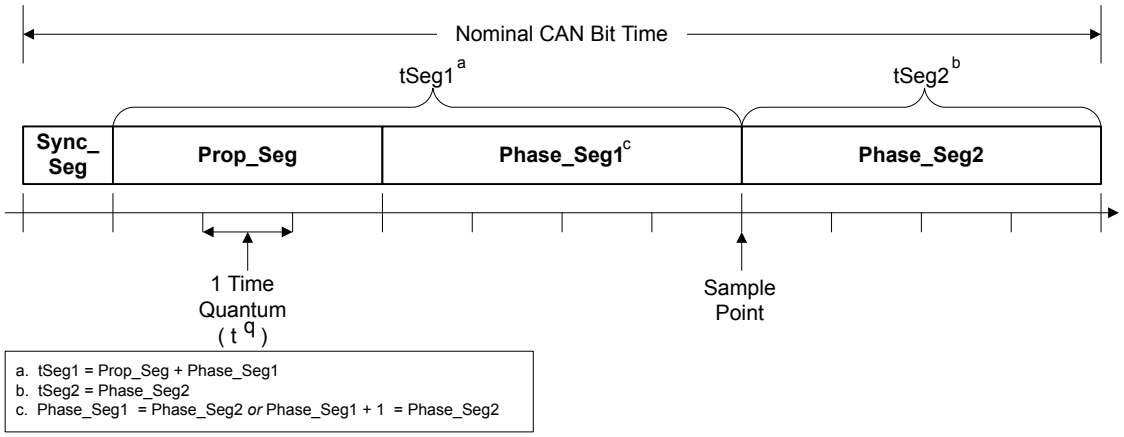


Table 17-1. CAN Protocol Ranges[a]

Parameter	Range	Remark
BRP	[1 .. 32]	Defines the length of the time quantum t_q
Sync_Seg	1 t_q	Fixed length, synchronization of bus input to system clock
Prop_Seg	[1 .. 8] t_q	Compensates for the physical delay times
Phase_Seg1	[1 .. 8] t_q	May be lengthened temporarily by synchronization
Phase_Seg2	[1 .. 8] t_q	May be shortened temporarily by synchronization
SJW	[1 .. 4] t_q	May not be longer than either Phase Buffer Segment

a. This table describes the minimum programmable ranges required by the CAN protocol.

The bit timing configuration is programmed in two register bytes in the **CANBIT** register. The sum of Prop_Seg and Phase_Seg1 (as TSEG1) is combined with Phase_Seg2 (as TSEG2) in one byte, and SJW and BRP are combined in the other byte.

In these bit timing registers, the four components TSEG1, TSEG2, SJW, and BRP have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1..n], values in the range of [0..n-1] are programmed. That way, for example, SJW (functional range of [1..4]) is represented by only two bits. Therefore, the length of the bit time is (programmed values):

$[TSEG1 + TSEG2 + 3] \times t_q$

or (functional values):

$[Sync\_Seg + Prop\_Seg + Phase\_Seg1 + Phase\_Seg2] \times t_q$

The data in the **CANBIT** register is the configuration input of the CAN protocol controller. The baud rate prescaler (configured by the BRP field) defines the length of the time quantum, the basic time unit of the bit time; the bit timing logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the sample point, and occasional synchronizations are controlled by the CAN controller and are evaluated once per time quantum.

The CAN controller translates messages to and from frames. It generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the

error management, and decides which type of synchronization is to be used. It is evaluated at the sample point and processes the sampled bus input bit. The time after the sample point that is needed to calculate the next bit to be sent (that is, the data bit, CRC bit, stuff bit, error flag, or idle) is called the information processing time (IPT).

The IPT is application-specific but may not be longer than 2 t_q; the CAN's IPT is 0 t_q. Its length is the lower limit of the programmed length of Phase_Seg2. In case of synchronization, Phase_Seg2 may be shortened to a value less than IPT, which does not affect bus timing.

17.2.16 Calculating the Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a required bit rate or bit time. The resulting bit time (1/bit rate) must be an integer multiple of the system clock period.

The bit time may consist of 4 to 25 time quanta. Several combinations may lead to the required bit time, allowing iterations of the following steps.

The first part of the bit time to be defined is the Prop_Seg. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandable CAN bus systems. The resulting time for Prop_Seg is converted into time quanta (rounded up to the nearest integer multiple of t_q).

The Sync_Seg is 1 t_q long (fixed), which leaves (bit time - Prop_Seg - 1) t_q for the two Phase Buffer Segments. If the number of remaining t_q is even, the Phase Buffer Segments have the same length, that is, Phase_Seg2 = Phase_Seg1, else Phase_Seg2 = Phase_Seg1 + 1.

The minimum nominal length of Phase_Seg2 has to be regarded as well. Phase_Seg2 may not be shorter than the CAN controller's IPT, which is t_q.

The length of the synchronization jump width is set to its maximum value, which is the minimum of 4 and Phase_Seg1.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formula given below:

$$(1 - df) \times fnom \leq fosc \leq (1 + df) \times fnom$$

where:

- df = Maximum tolerance of oscillator frequency
- fosc = Actual oscillator frequency
- fnom = Nominal oscillator frequency

Maximum frequency tolerance must take into account the following formulas:

$$df \leq \frac{(Phase\_Seg1, Phase\_Seg2)\ \min}{2 \times (13 \times tbit - Phase\_Seg2)}$$

$$df\max = 2 \times df \times fnom$$

where:

- Phase_Seg1 and Phase_Seg2 are from Table 17-1 on page 529

- tbit = Bit Time
- dfmax = Maximum difference between two oscillators

If more than one configuration is possible, that configuration allowing the highest oscillator tolerance range should be chosen.

CAN nodes with different system clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay times, is done once for the whole network.

The CAN system's oscillator tolerance range is limited by the node with the lowest tolerance range.

The calculation may show that bus length or bit rate have to be decreased or that the oscillator frequencies' stability has to be increased in order to find a protocol-compliant configuration of the CAN bit timing.

17.2.16.1 Example for Bit Timing at High Baud Rate

In this example, the frequency of CAN clock is 25 MHz, BRP is 4, and the bit rate is 1 Mbps.

t_q 200 ns = (BRP + 1)/CAN Clock
delay of bus driver 50 ns
delay of receiver circuit 30 ns
delay of bus line (40m) 220 ns
tProp 400 ns = 2 × t_q
tSJW 200 ns = 1 × t_q
tTSeg1 600 ns = tProp + tSJW
tTSeg2 200 ns = (Information Processing Time + 1) × t_q
tSync-Seg 200 ns = 1 × t_q
bit time 1000 ns = tSync-Seg + tTSeg1 + tTSeg2

In the above example, the bit field values for the **CANBIT** register are: TSEG2=1, TSEG1=2, SJW =3 and BRP=0. This makes the final value programmed into the **CANBIT** register = 0x3FC0.

17.2.16.2 Example for Bit Timing at Low Baud Rate

In this example, the frequency of the CAN clock is 50 MHz, BRP is 24, and the bit rate is 100 Kbps.

t_q 500 ns = (BRP + 1)/CAN clock
delay of bus driver 200 ns
delay of receiver circuit 80 ns
delay of bus line (40m) 220 ns
tProp 4.5 µs = 9 × t_q
tSJW 2 µs = 4 × t_q
tTSeg1 6.5 µs = tProp + tSJW
tTSeg2 3 µs = (Information Processing Time + 6) × t_q
tSync-Seg 500 ns = 1 × t_q
bit time 10 µs = tSync-Seg + tTSeg1 + tTSeg2

In the above example, the bit field values for the **CANBIT** register are: TSEG2=5, TSEG1=12, SJW =3 and BRP=24. This makes the final value programmed into the **CANBIT** register = 0x5CD8.

17.3 Register Map

Table 17-2 on page 532 lists the registers. All addresses given are relative to the CAN base address of:

- CAN0: 0x4004.0000
- CAN1: 0x4004.1000

Table 17-2. CAN Register Map

Offset	Name	Type	Reset	Description	See page
0x000	CANCTL	R/W	0x0000.0001	CAN Control	534
0x004	CANSTS	R/W	0x0000.0000	CAN Status	536
0x008	CANERR	RO	0x0000.0000	CAN Error Counter	539
0x00C	CANBIT	R/W	0x0000.2301	CAN Bit Timing	540
0x010	CANINT	RO	0x0000.0000	CAN Interrupt	542
0x014	CANTST	R/W	0x0000.0000	CAN Test	543
0x018	CANBRPE	R/W	0x0000.0000	CAN Baud Rate Prescaler Extension	545
0x020	CANIF1CRQ	R/W	0x0000.0001	CAN IF1 Command Request	546
0x024	CANIF1CMSK	R/W	0x0000.0000	CAN IF1 Command Mask	547
0x028	CANIF1MSK1	R/W	0x0000.FFFF	CAN IF1 Mask 1	549
0x02C	CANIF1MSK2	R/W	0x0000.FFFF	CAN IF1 Mask 2	550
0x030	CANIF1ARB1	R/W	0x0000.0000	CAN IF1 Arbitration 1	551
0x034	CANIF1ARB2	R/W	0x0000.0000	CAN IF1 Arbitration 2	552
0x038	CANIF1MCTL	R/W	0x0000.0000	CAN IF1 Message Control	554
0x03C	CANIF1DA1	R/W	0x0000.0000	CAN IF1 Data A1	556
0x040	CANIF1DA2	R/W	0x0000.0000	CAN IF1 Data A2	556
0x044	CANIF1DB1	R/W	0x0000.0000	CAN IF1 Data B1	556
0x048	CANIF1DB2	R/W	0x0000.0000	CAN IF1 Data B2	556
0x080	CANIF2CRQ	R/W	0x0000.0001	CAN IF2 Command Request	546
0x084	CANIF2CMSK	R/W	0x0000.0000	CAN IF2 Command Mask	547
0x088	CANIF2MSK1	R/W	0x0000.FFFF	CAN IF2 Mask 1	549
0x08C	CANIF2MSK2	R/W	0x0000.FFFF	CAN IF2 Mask 2	550
0x090	CANIF2ARB1	R/W	0x0000.0000	CAN IF2 Arbitration 1	551
0x094	CANIF2ARB2	R/W	0x0000.0000	CAN IF2 Arbitration 2	552
0x098	CANIF2MCTL	R/W	0x0000.0000	CAN IF2 Message Control	554
0x09C	CANIF2DA1	R/W	0x0000.0000	CAN IF2 Data A1	556
0x0A0	CANIF2DA2	R/W	0x0000.0000	CAN IF2 Data A2	556

Offset	Name	Type	Reset	Description	See page
0x0A4	CANIF2DB1	R/W	0x0000.0000	CAN IF2 Data B1	556
0x0A8	CANIF2DB2	R/W	0x0000.0000	CAN IF2 Data B2	556
0x100	CANTXRQ1	RO	0x0000.0000	CAN Transmission Request 1	557
0x104	CANTXRQ2	RO	0x0000.0000	CAN Transmission Request 2	557
0x120	CANNWDA1	RO	0x0000.0000	CAN New Data 1	558
0x124	CANNWDA2	RO	0x0000.0000	CAN New Data 2	558
0x140	CANMSG1INT	RO	0x0000.0000	CAN Message 1 Interrupt Pending	559
0x144	CANMSG2INT	RO	0x0000.0000	CAN Message 2 Interrupt Pending	559
0x160	CANMSG1VAL	RO	0x0000.0000	CAN Message 1 Valid	560
0x164	CANMSG2VAL	RO	0x0000.0000	CAN Message 2 Valid	560

17.4 CAN Register Descriptions

The remainder of this section lists and describes the CAN registers, in numerical order by address offset. There are two sets of Interface Registers that are used to access the Message Objects in the Message RAM: **CANIF1x** and **CANIF2x**. The function of the two sets are identical and are used to queue transactions.

Register 1: CAN Control (CANCTL), offset 0x000

This control register initializes the module and enables test mode and interrupts.

The bus-off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or clearing INIT. If the device goes bus-off, it sets INIT, stopping all bus activities. Once INIT has been cleared by the CPU, the device then waits for 129 occurrences of Bus Idle (129 * 11 consecutive High bits) before resuming normal operations. At the end of the bus-off recovery sequence, the Error Management Counters are reset.

During the waiting time after INIT is cleared, each time a sequence of 11 High bits has been monitored, a BITERROR0 code is written to the **CANSTS** register (the LEC field = 0x5), enabling the CPU to readily check whether the CAN bus is stuck Low or continuously disturbed, and to monitor the proceeding of the bus-off recovery sequence.

CAN Control (CANCTL)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x000
Type R/W, reset 0x0000.0001

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Field	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	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	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	reserved								TEST	CCE	DAR	reserved	EIE	SIE	IE	INIT
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TEST	R/W	0	Test Mode Enable 0: Normal operation 1: Test mode
6	CCE	R/W	0	Configuration Change Enable 0: Do not allow write access to the **CANBIT** register. 1: Allow write access to the **CANBIT** register if the INIT bit is 1.
5	DAR	R/W	0	Disable Automatic-Retransmission 0: Auto-retransmission of disturbed messages is enabled. 1: Auto-retransmission is disabled.
4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
3	EIE	R/W	0	Error Interrupt Enable 0: Disabled. No error status interrupt is generated. 1: Enabled. A change in the BOFF or EWARN bits in the **CANSTS** register generates an interrupt.
2	SIE	R/W	0	Status Interrupt Enable 0: Disabled. No status interrupt is generated. 1: Enabled. An interrupt is generated when a message has successfully been transmitted or received, or a CAN bus error has been detected. A change in the TXOK, RXOK or LEC bits in the **CANSTS** register generates an interrupt.
1	IE	R/W	0	CAN Interrupt Enable 0: Interrupts disabled. 1: Interrupts enabled.
0	INIT	R/W	1	Initialization 0: Normal operation. 1: Initialization started.

Register 2: CAN Status (CANSTS), offset 0x004

The status register contains information for interrupt servicing such as Bus-Off, error count threshold, and error types.

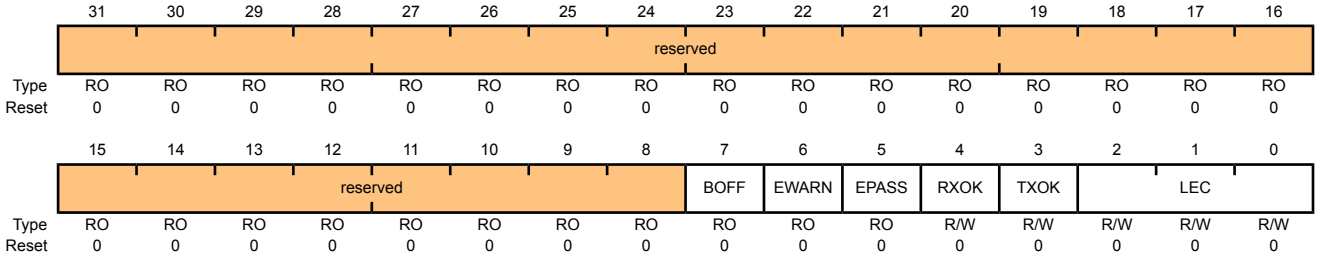
The LEC field holds the code that indicates the type of the last error to occur on the CAN bus. This field is cleared when a message has been transferred (reception or transmission) without error. The unused error code 7 may be written by the CPU to manually set this field to an invalid error so that it can be checked for a change later.

An error interrupt is generated by the BOFF and EWARN bits and a status interrupt is generated by the RXOK, TXOK, and LEC bits, if the corresponding enable bits in the **CAN Control (CANCTL)** register are set. A change of the EPASS bit or a write to the RXOK, TXOK, or LEC bits does not generate an interrupt.

Reading the **CAN Status (CANSTS)** register clears the **CAN Interrupt (CANINT)** register, if it is pending.

CAN Status (CANSTS)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x004
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	BOFF	RO	0	Bus-Off Status 0: CAN controller is not in bus-off state. 1: CAN controller is in bus-off state.
6	EWARN	RO	0	Warning Status 0: Both error counters are below the error warning limit of 96. 1: At least one of the error counters has reached the error warning limit of 96.
5	EPASS	RO	0	Error Passive 0: The CAN module is in the Error Active state, that is, the receive or transmit error count is less than or equal to 127. 1: The CAN module is in the Error Passive state, that is, the receive or transmit error count is greater than 127.

Bit/Field	Name	Type	Reset	Description
4	RXOK	R/W	0	Received a Message Successfully 0: Since this bit was last cleared, no message has been successfully received. 1: Since this bit was last cleared, a message has been successfully received, independent of the result of the acceptance filtering. This bit is never cleared by the CAN module.
3	TXOK	R/W	0	Transmitted a Message Successfully 0: Since this bit was last cleared, no message has been successfully transmitted. 1: Since this bit was last cleared, a message has been successfully transmitted error-free and acknowledged by at least one other node. This bit is never cleared by the CAN module.

Bit/Field	Name	Type	Reset	Description
2:0	LEC	R/W	0x0	Last Error Code

This is the type of the last error to occur on the CAN bus.

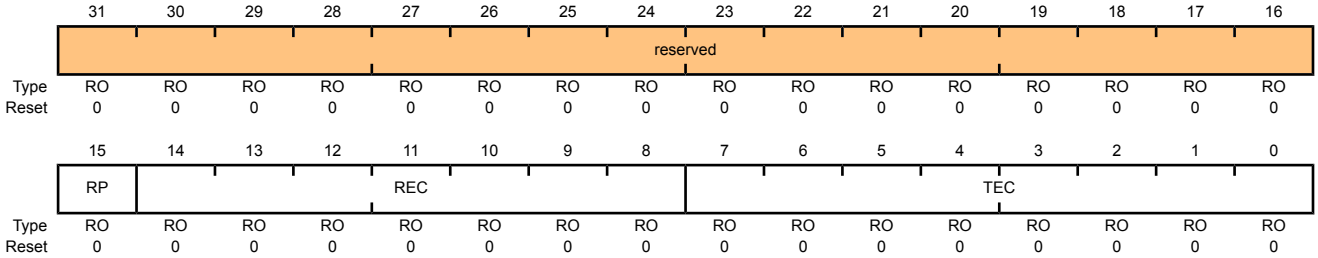
Value	Definition
0x0	No Error
0x1	Stuff Error More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
0x2	Format Error A fixed format part of the received frame has the wrong format.
0x3	ACK Error The message transmitted was not acknowledged by another node.
0x4	Bit 1 Error When a message is transmitted, the CAN controller monitors the data lines to detect any conflicts. When the arbitration field is transmitted, data conflicts are a part of the arbitration protocol. When other frame fields are transmitted, data conflicts are considered errors. A Bit 1 Error indicates that the device wanted to send a High level (logical 1) but the monitored bus value was Low (logical 0).
0x5	Bit 0 Error A Bit 0 Error indicates that the device wanted to send a Low level (logical 0), but the monitored bus value was High (logical 1). During bus-off recovery, this status is set each time a sequence of 11 High bits has been monitored. This enables the CPU to monitor the proceeding of the bus-off recovery sequence without any disturbances to the bus.
0x6	CRC Error The CRC checksum was incorrect in the received message, indicating that the calculated value received did not match the calculated CRC of the data.
0x7	Unused When the LEC bit shows this value, no CAN bus event was detected since the CPU wrote this value to LEC.

Register 3: CAN Error Counter (CANERR), offset 0x008

This register contains the error counter values, which can be used to analyze the cause of an error.

CAN Error Counter (CANERR)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x008
Type RO, reset 0x0000.0000



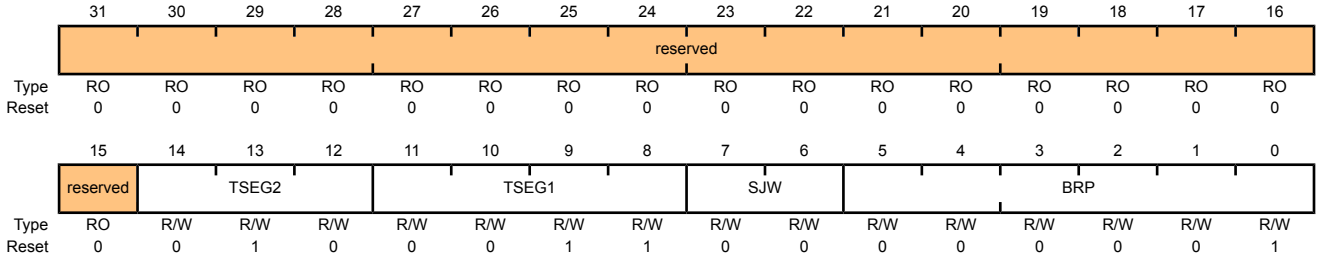
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	RP	RO	0	Received Error Passive 0: The Receive Error counter is below the Error Passive level (127 or less). 1: The Receive Error counter has reached the Error Passive level (128 or greater).
14:8	REC	RO	0x00	Receive Error Counter State of the receiver error counter (0 to 127).
7:0	TEC	RO	0x00	Transmit Error Counter State of the transmit error counter (0 to 255).

Register 4: CAN Bit Timing (CANBIT), offset 0x00C

This register is used to program the bit width and bit quantum. Values are programmed to the system clock frequency. This register is write-enabled by setting the CCE and INIT bits in the **CANCTL** register. See “Bit Time and Bit Rate” on page 528 for more information.

CAN Bit Timing (CANBIT)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x00C
Type R/W, reset 0x0000.2301



Bit/Field	Name	Type	Reset	Description
31:15	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
14:12	TSEG2	R/W	0x2	Time Segment after Sample Point 0x00-0x07: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. So, for example, a reset value of 0x2 defines that there is 3 (2+1) bit time quanta defined for Phase_Seg2 (see Figure 17-4 on page 529). The bit time quanta is defined by the BRP field.
11:8	TSEG1	R/W	0x3	Time Segment Before Sample Point 0x00-0x0F: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. So, for example, the reset value of 0x3 defines that there is 4 (3+1) bit time quanta defined for Phase_Seg1 (see Figure 17-4 on page 529). The bit time quanta is define by the BRP field.
7:6	SJW	R/W	0x0	(Re)Synchronization Jump Width 0x00-0x03: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. During the start of frame (SOF), if the CAN controller detects a phase error (misalignment), it can adjust the length of TSEG2 or TSEG1 by the value in SJW. So the reset value of 0 adjusts the length by 1 bit time quanta.

Bit/Field	Name	Type	Reset	Description
5:0	BRP	R/W	0x1	Baud Rate Prescaler The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quantum. 0x00-0x03F: The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. BRP defines the number of CAN clock periods that make up 1 bit time quanta, so the reset value is 2 bit time quanta (1+1). The **CANBRPE** register can be used to further divide the bit time.

Register 5: CAN Interrupt (CANINT), offset 0x010

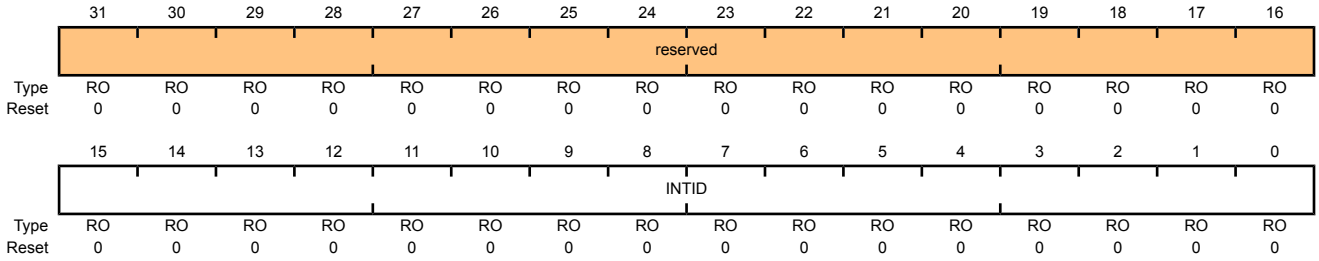
This register indicates the source of the interrupt.

If several interrupts are pending, the **CAN Interrupt (CANINT)** register points to the pending interrupt with the highest priority, disregarding the order in which the interrupts occurred. An interrupt remains pending until the CPU has cleared it. If the `INTID` field is not 0x0000 (the default) and the `IE` bit in the **CANCTL** register is set, the interrupt is active. The interrupt line remains active until the `INTID` field is cleared by reading the **CANSTS** register, or until the `IE` bit in the **CANCTL** register is cleared.

Note: Reading the **CAN Status (CANSTS)** register clears the **CAN Interrupt (CANINT)** register, if it is pending.

CAN Interrupt (CANINT)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x010
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	INTID	RO	0x0000	Interrupt Identifier The number in this field indicates the source of the interrupt.

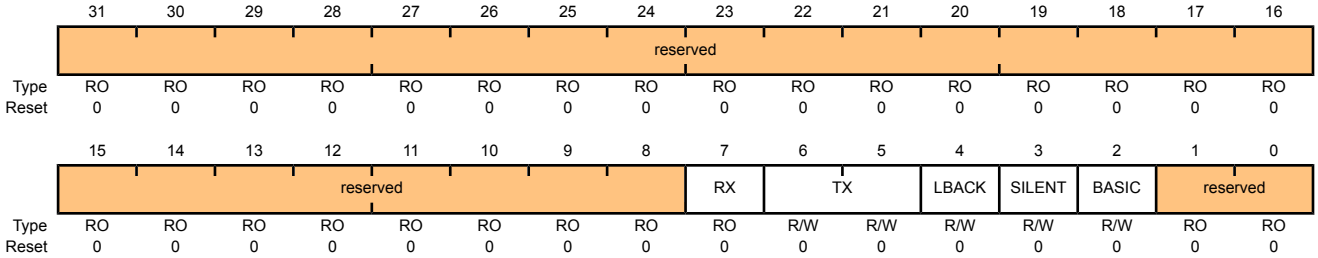
Value	Definition
0x0000	No interrupt pending
0x0001-0x0020	Number of the message object that caused the interrupt
0x0021-0x7FFF	Unused
0x8000	Status Interrupt
0x8001-0xFFFF	Unused

Register 6: CAN Test (CANTST), offset 0x014

This is the test mode register for self-test and external pin access. It is write-enabled by setting the TEST bit in the **CANCTL** register. Different test functions may be combined, however, CAN transfers will be affected if the TX bits in this register are not zero.

CAN Test (CANTST)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x014
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	RX	RO	0	Receive Observation Displays the value on the CANnRx pin.
6:5	TX	R/W	0x0	Transmit Control Overrides control of the CANnTx pin. Value Description 0x0 CANnTx is controlled by the CAN module; default operation 0x1 The sample point is driven on the CANnTx signal. This mode is useful to monitor bit timing. 0x2 CANnTx drives a low value. This mode is useful for checking the physical layer of the CAN bus. 0x3 CANnTx drives a high value. This mode is useful for checking the physical layer of the CAN bus.
4	LBACK	R/W	0	Loopback Mode 0: Disabled. 1: Enabled. In loopback mode, the data from the transmitter is routed into the receiver. Any data on the receive input is ignored.
3	SILENT	R/W	0	Silent Mode Do not transmit data; monitor the bus. Also known as Bus Monitor mode. 0: Disabled. 1: Enabled.

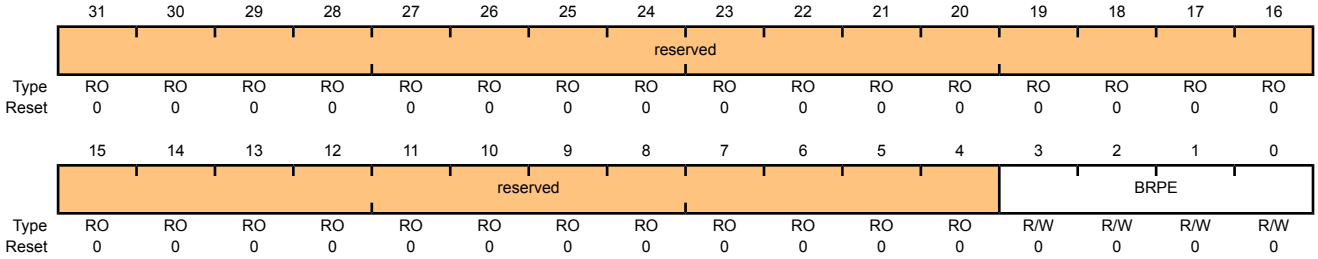
Bit/Field	Name	Type	Reset	Description
2	BASIC	R/W	0	Basic Mode 0: Disabled. 1: Use **CANIF1** registers as transmit buffer, and use **CANIF2** registers as receive buffer.
1:0	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 7: CAN Baud Rate Prescaler Extension (CANBRPE), offset 0x018

This register is used to further divide the bit time set with the BRP bit in the **CANBIT** register. It is write-enabled by setting the CCE bit in the **CANCTL** register.

CAN Baud Rate Prescaler Extension (CANBRPE)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x018
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x0000.000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	BRPE	R/W	0x0	Baud Rate Prescaler Extension 0x00-0x0F: Extend the BRP bit in the **CANBIT** register to values up to 1023. The actual interpretation by the hardware is one more than the value programmed by BRPE (MSBs) and BRP (LSBs).

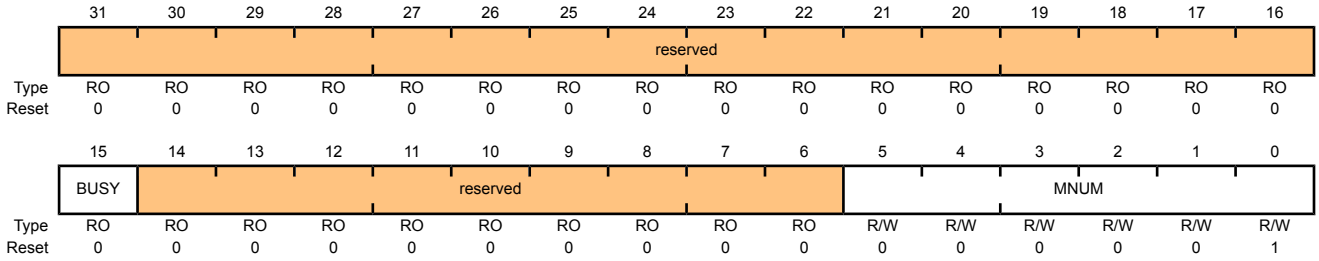
Register 8: CAN IF1 Command Request (CANIF1CRQ), offset 0x020

Register 9: CAN IF2 Command Request (CANIF2CRQ), offset 0x080

A message transfer is started as soon as there is a write of the message object number to the MNUM field when the TXRQST bit in the **CANIF1MCTL** register is set. With this write operation, the BUSY bit is automatically set to indicate that a transfer between the CAN Interface Registers and the internal message RAM is in progress. After a wait time of 3 to 6 CAN_CLK periods, the transfer between the interface register and the message RAM completes, which then clears the BUSY bit.

CAN IF1 Command Request (CANIF1CRQ)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x020
Type R/W, reset 0x0000.0001



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	BUSY	RO	0	Busy Flag 0: Cleared when read/write action has finished. 1: Set when a write occurs to the message number in this register.
14:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5:0	MNUM	R/W	0x01	Message Number Selects one of the 32 message objects in the message RAM for data transfer. The message objects are numbered from 1 to 32.

Value	Description
0x00	0 is not a valid message number; it is interpreted as 0x20, or object 32.
0x01-0x20	Indicates specified message object 1 to 32.
0x21-0x3F	Not a valid message number; values are shifted and it is interpreted as 0x01-0x1F.

Register 10: CAN IF1 Command Mask (CANIF1CMSK), offset 0x024

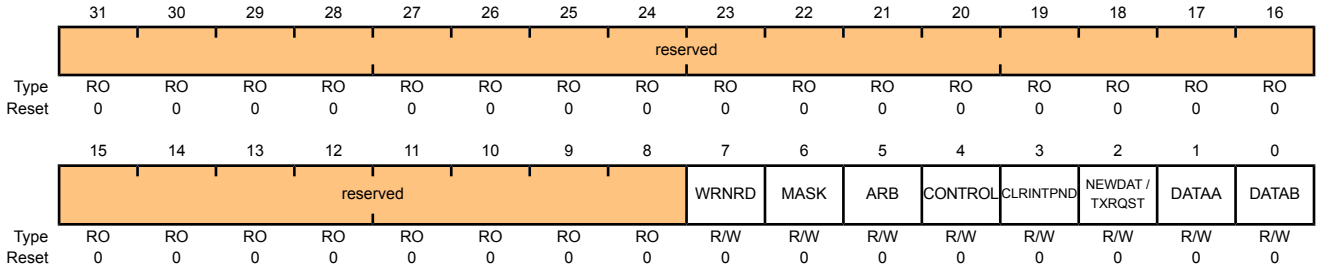
Register 11: CAN IF2 Command Mask (CANIF2CMSK), offset 0x084

Reading the Command Mask registers provides status for various functions. Writing to the Command Mask registers specifies the transfer direction and selects which buffer registers are the source or target of the data transfer.

Note that when a read from the message object buffer occurs when the WRNRD bit is clear and the CLRINTPND and/or NEWDAT bits are set, the interrupt pending and/or new data flags in the message object buffer are cleared.

CAN IF1 Command Mask (CANIF1CMSK)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x024
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x0000.00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	WRNRD	R/W	0	Write, Not Read Transfer the message object address specified by the **CAN Command Request (CANIFnCRQ)** register to the CAN message buffer registers. **Note:** Interrupt pending and new data conditions in the message buffer can be cleared by reading from the buffer (WRNRD = 0) when the CLRINTPND and/or NEWDAT bits are set.
6	MASK	R/W	0	Access Mask Bits 0: Mask bits unchanged. 1: Transfer IDMASK + DIR + MXTD of the message object into the Interface registers.
5	ARB	R/W	0	Access Arbitration Bits 0: Arbitration bits unchanged. 1: Transfer ID + DIR + XTD + MSGVAL of the message object into the Interface registers.
4	CONTROL	R/W	0	Access Control Bits 0: Control bits unchanged. 1: Transfer control bits from the **CANIFnMCTL** register into the Interface registers.

Bit/Field	Name	Type	Reset	Description
3	CLRINTPND	R/W	0	Clear Interrupt Pending Bit If WRNRD is set, this bit controls whether the INTPND bit in the **CANIFnMCTL** register is changed. 0: The INTPND bit in the message object remains unchanged. 1: The INTPND bit is cleared in the message object. If WRNRD is clear and this bit is clear, the interrupt pending status is transferred from the message buffer into the **CANIFnMCTL** register. If WRNRD is clear and this bit is set, the interrupt pending status is cleared in the message buffer. Note that the value of this bit that is transferred to the **CANIFnMCTL** register always reflects the status of the bits before clearing.
2	NEWDAT / TXRQST	R/W	0	NEWDAT / TXRQST Bit If WRNRD is set, this bit can act as a TXRQST bit and request a transmission. Note that when this bit is set, the TXRQST bit in the **CANIFnMCTL** register is ignored. 0: Transmission is not requested 1: Begin a transmission If WRNRD is clear and this bit is clear, the value of the new data status is transferred from the message buffer into the **CANIFnMCTL** register. If WRNRD is clear and this bit is set, the new data status is cleared in the message buffer. Note that the value of this bit that is transferred to the **CANIFnMCTL** register always reflects the status of the bits before clearing.
1	DATAA	R/W	0	Access Data Byte 0 to 3 When WRNRD = 1: 0: Data bytes 0-3 are unchanged. 1: Transfer data bytes 0-3 in message object to **CANIFnDA1** and **CANIFnDA2**. When WRNRD = 0: 0: Data bytes 0-3 are unchanged. 1: Transfer data bytes 0-3 in **CANIFnDA1** and **CANIFnDA2** to the message object.
0	DATAB	R/W	0	Access Data Byte 4 to 7 When WRNRD = 1: 0: Data bytes 4-7 are unchanged. 1: Transfer data bytes 4-7 in message object to **CANIFnDB1** and **CANIFnDB2**. When WRNRD = 0: 0: Data bytes 4-7 are unchanged. 1: Transfer data bytes 4-7 in **CANIFnDB1** and **CANIFnDB2** to the message object.

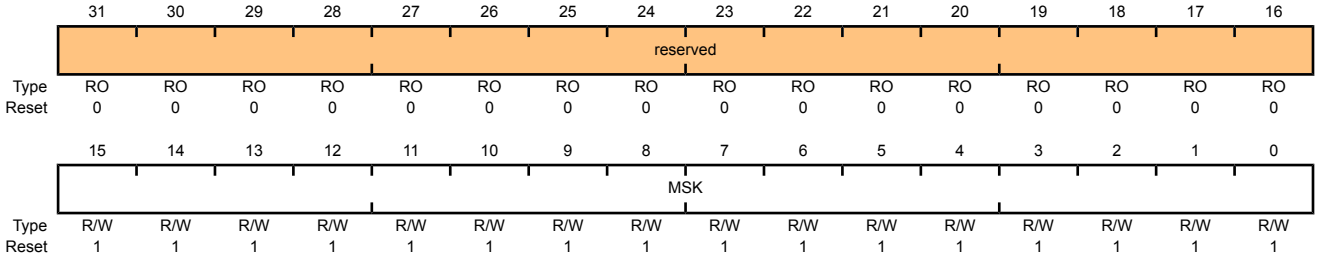
Register 12: CAN IF1 Mask 1 (CANIF1MSK1), offset 0x028

Register 13: CAN IF2 Mask 1 (CANIF2MSK1), offset 0x088

The mask information provided in this register accompanies the data (**CANIFnDAn**), arbitration information (**CANIFnARBn**), and control information (**CANIFnMCTL**) to the message object in the message RAM. The mask is used with the `ID` bit in the **CANIFnARBn** register for acceptance filtering. Additional mask information is contained in the **CANIFnMSK2** register.

CAN IF1 Mask 1 (CANIF1MSK1)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x028
Type R/W, reset 0x0000.FFFF



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MSK	R/W	0xFFFF	Identifier Mask When using a 29-bit identifier, these bits are used for bits [15:0] of the ID. The `MSK` field in the **CANIFnMSK2** register are used for bits [28:16] of the ID. When using an 11-bit identifier, these bits are ignored. 0: The corresponding identifier field (`ID`) in the message object cannot inhibit the match in acceptance filtering. 1: The corresponding identifier field (`ID`) is used for acceptance filtering.

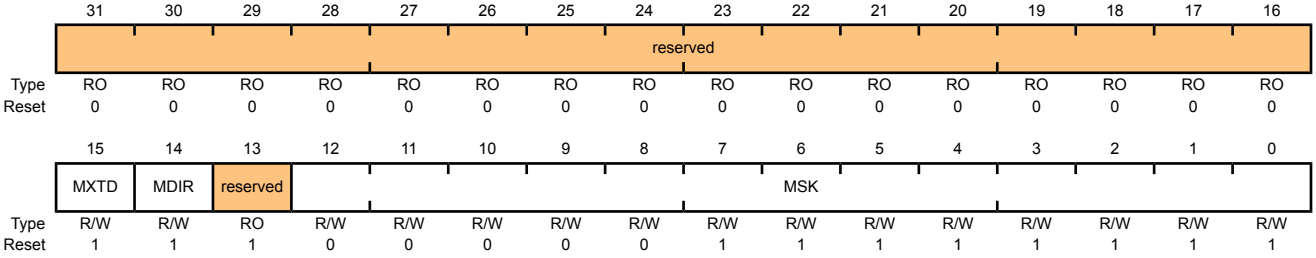
Register 14: CAN IF1 Mask 2 (CANIF1MSK2), offset 0x02C

Register 15: CAN IF2 Mask 2 (CANIF2MSK2), offset 0x08C

This register holds extended mask information that accompanies the **CANIFnMSK1** register.

CAN IF1 Mask 2 (CANIF1MSK2)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x02C
Type R/W, reset 0x0000.FFFF



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	MXTD	R/W	0x1	Mask Extended Identifier 0: The extended identifier bit (`XTD` in the **CANIFnARB2** register) has no effect on the acceptance filtering. 1: The extended identifier bit `XTD` is used for acceptance filtering.
14	MDIR	R/W	0x1	Mask Message Direction 0: The message direction bit (`DIR` in the **CANIFnARB2** register) has no effect for acceptance filtering. 1: The message direction bit `DIR` is used for acceptance filtering.
13	reserved	RO	0x1	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12:0	MSK	R/W	0xFF	Identifier Mask When using a 29-bit identifier, these bits are used for bits [28:16] of the ID. The `MSK` field in the **CANIFnMSK1** register are used for bits [15:0] of the ID. When using an 11-bit identifier, `MSK[12:2]` are used for bits [10:0] of the ID. 0: The corresponding identifier field (`ID`) in the message object cannot inhibit the match in acceptance filtering. 1: The corresponding identifier field (`ID`) is used for acceptance filtering.

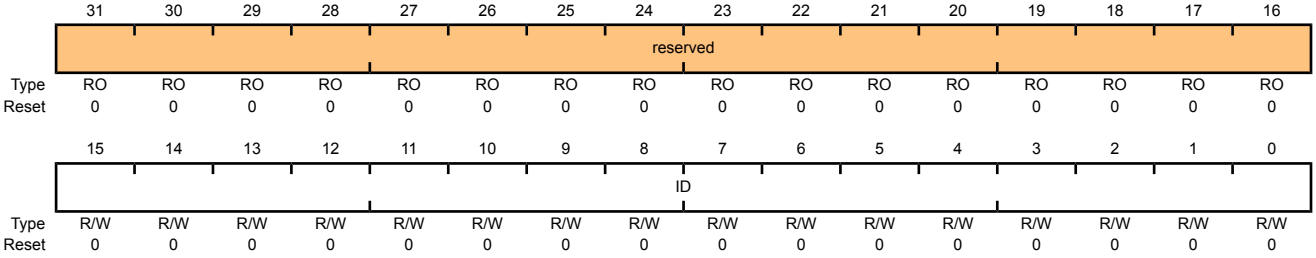
Register 16: CAN IF1 Arbitration 1 (CANIF1ARB1), offset 0x030

Register 17: CAN IF2 Arbitration 1 (CANIF2ARB1), offset 0x090

These registers hold the identifiers for acceptance filtering.

CAN IF1 Arbitration 1 (CANIF1ARB1)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x030
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	ID	R/W	0x0000	Message Identifier This bit field is used with the `ID` field in the **CANIFnARB2** register to create the message identifier. When using a 29-bit identifier, bits 15:0 of the **CANIFnARB1** register are [15:0] of the ID, while bits 12:0 of the **CANIFnARB2** register are [28:16] of the ID. When using an 11-bit identifier, these bits are not used.

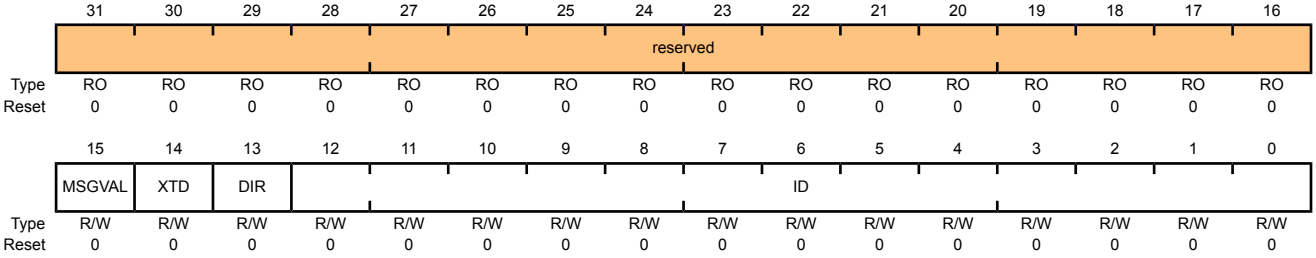
Register 18: CAN IF1 Arbitration 2 (CANIF1ARB2), offset 0x034

Register 19: CAN IF2 Arbitration 2 (CANIF2ARB2), offset 0x094

These registers hold information for acceptance filtering.

CAN IF1 Arbitration 2 (CANIF1ARB2)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x034
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	MSGVAL	R/W	0	Message Valid 0: The message object is ignored by the message handler. 1: The message object is configured and ready to be considered by the message handler within the CAN controller. All unused message objects should have this bit cleared during initialization and before clearing the `INIT` bit in the **CANCTL** register. The `MSGVAL` bit must also be cleared before any of the following bits are modified or if the message object is no longer required: the `ID` fields in the **CANIFnARBn** registers, the `XTD` and `DIR` bits in the **CANIFnARB2** register, or the `DLC` field in the **CANIFnMCTL** register.
14	XTD	R/W	0	Extended Identifier 0: An 11-bit Standard Identifier is used for this message object. 1: A 29-bit Extended Identifier is used for this message object.
13	DIR	R/W	0	Message Direction 0: Receive. When the `TXRQST` bit in the **CANIFnMCTL** register is set, a remote frame with the identifier of this message object is received. On reception of a data frame with matching identifier, that message is stored in this message object. 1: Transmit. When the `TXRQST` bit in the **CANIFnMCTL** register is set, the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, the `TXRQST` bit of this message object is set (if `RMTEN=1`).

Bit/Field	Name	Type	Reset	Description
12:0	ID	R/W	0x000	Message Identifier This bit field is used with the `ID` field in the **CANIFnARB2** register to create the message identifier. When using a 29-bit identifier, `ID[15:0]` of the **CANIFnARB1** register are [15:0] of the ID, while these bits, `ID[12:0]`, are [28:16] of the ID. When using an 11-bit identifier, `ID[12:2]` are used for bits [10:0] of the ID. The `ID` field in the **CANIFnARB1** register is ignored.

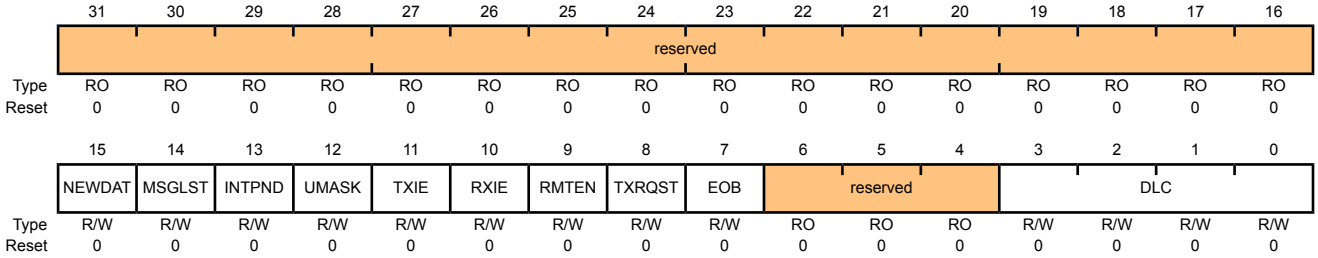
Register 20: CAN IF1 Message Control (CANIF1MCTL), offset 0x038

Register 21: CAN IF2 Message Control (CANIF2MCTL), offset 0x098

This register holds the control information associated with the message object to be sent to the Message RAM.

CAN IF1 Message Control (CANIF1MCTL)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x038
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15	NEWDAT	R/W	0	New Data 0: No new data has been written into the data portion of this message object by the message handler since the last time this flag was cleared by the CPU. 1: The message handler or the CPU has written new data into the data portion of this message object.
14	MSGLST	R/W	0	Message Lost 0 : No message was lost since the last time this bit was cleared by the CPU. 1: The message handler stored a new message into this object when NEWDAT was set; the CPU has lost a message. This bit is only valid for message objects when the DIR bit in the **CANIFnARB2** register clear (receive).
13	INTPND	R/W	0	Interrupt Pending 0: This message object is not the source of an interrupt. 1: This message object is the source of an interrupt. The interrupt identifier in the **CANINT** register points to this message object if there is not another interrupt source with a higher priority.
12	UMASK	R/W	0	Use Acceptance Mask 0: Mask ignored. 1: Use mask (MSK, MXTD, and MDIR bits in the **CANIFnMSKn** registers) for acceptance filtering.

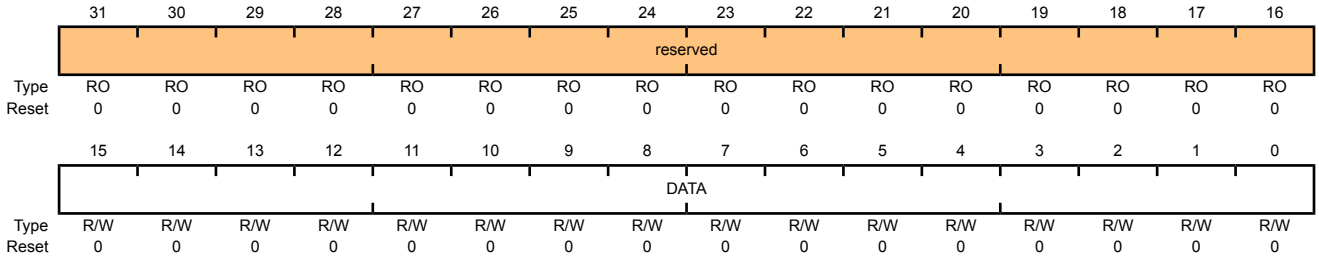
Bit/Field	Name	Type	Reset	Description
11	TXIE	R/W	0	Transmit Interrupt Enable 0: The INTPND bit in the **CANIFnMCTL** register is unchanged after a successful transmission of a frame. 1: The INTPND bit in the **CANIFnMCTL** register is set after a successful transmission of a frame.
10	RXIE	R/W	0	Receive Interrupt Enable 0: The INTPND bit in the **CANIFnMCTL** register is unchanged after a successful reception of a frame. 1: The INTPND bit in the **CANIFnMCTL** register is set after a successful reception of a frame.
9	RMTEN	R/W	0	Remote Enable 0: At the reception of a remote frame, the TXRQST bit in the **CANIFnMCTL** register is left unchanged. 1: At the reception of a remote frame, the TXRQST bit in the **CANIFnMCTL** register is set.
8	TXRQST	R/W	0	Transmit Request 0: This message object is not waiting for transmission. 1: The transmission of this message object is requested and is not yet done.
7	EOB	R/W	0	End of Buffer 0: Message object belongs to a FIFO Buffer and is not the last message object of that FIFO Buffer. 1: Single message object or last message object of a FIFO Buffer. This bit is used to concatenate two or more message objects (up to 32) to build a FIFO buffer. For a single message object (thus not belonging to a FIFO buffer), this bit must be set.
6:4	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	DLC	R/W	0x0	Data Length Code Value Description 0x0-0x8 Specifies the number of bytes in the data frame. 0x9-0xF Defaults to a data frame with 8 bytes. The DLC field in the **CANIFnMCTL** register of a message object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the message handler stores a data frame, it writes DLC to the value given by the received message.

Register 22: CAN IF1 Data A1 (CANIF1DA1), offset 0x03C
Register 23: CAN IF1 Data A2 (CANIF1DA2), offset 0x040
Register 24: CAN IF1 Data B1 (CANIF1DB1), offset 0x044
Register 25: CAN IF1 Data B2 (CANIF1DB2), offset 0x048
Register 26: CAN IF2 Data A1 (CANIF2DA1), offset 0x09C
Register 27: CAN IF2 Data A2 (CANIF2DA2), offset 0x0A0
Register 28: CAN IF2 Data B1 (CANIF2DB1), offset 0x0A4
Register 29: CAN IF2 Data B2 (CANIF2DB2), offset 0x0A8

These registers contain the data to be sent or that has been received. In a CAN data frame, data byte 0 is the first byte to be transmitted or received and data byte 7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte is transmitted first.

CAN IF1 Data A1 (CANIF1DA1)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x03C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	DATA	R/W	0x0000	Data The **CANIFnDA1** registers contain data bytes 1 and 0; **CANIFnDA2** data bytes 3 and 2; **CANIFnDB1** data bytes 5 and 4; and **CANIFnDB2** data bytes 7 and 6.

Register 30: CAN Transmission Request 1 (CANTXRQ1), offset 0x100

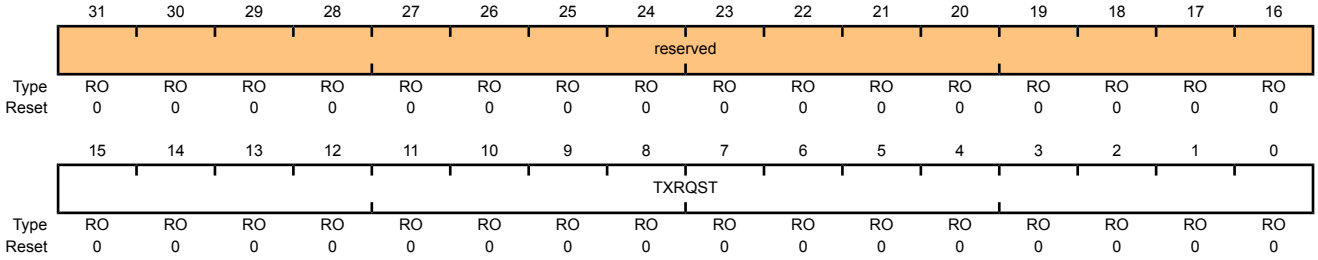
Register 31: CAN Transmission Request 2 (CANTXRQ2), offset 0x104

The **CANTXRQ1** and **CANTXRQ2** registers hold the TXRQST bits of the 32 message objects. By reading out these bits, the CPU can check which message object has a transmission request pending. The TXRQST bit of a specific message object can be changed by three sources: (1) the CPU via the **CANIFnMCTL** register, (2) the message handler state machine after the reception of a remote frame, or (3) the message handler state machine after a successful transmission.

The **CANTXRQ1** register contains the TXRQST bits of the first 16 message objects in the message RAM; the **CANTXRQ2** register contains the TXRQST bits of the second 16 message objects.

CAN Transmission Request 1 (CANTXRQ1)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x100
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	TXRQST	RO	0x0000	Transmission Request Bits 0: The corresponding message object is not waiting for transmission. 1: The transmission of the corresponding message object is requested and is not yet done.

Register 32: CAN New Data 1 (CANNWDA1), offset 0x120

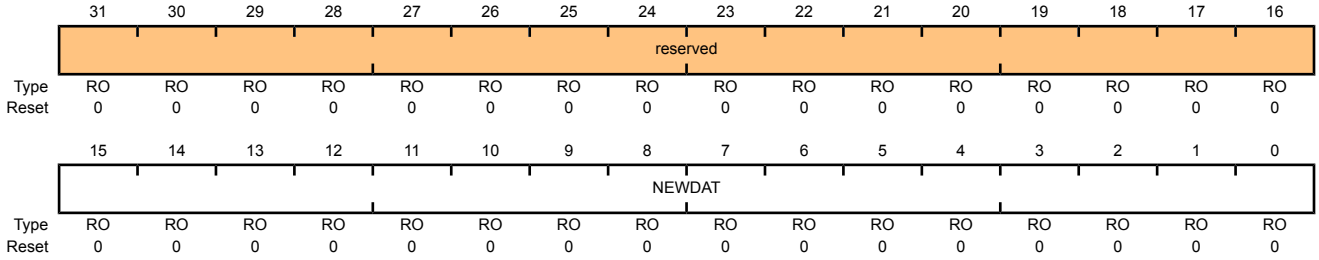
Register 33: CAN New Data 2 (CANNWDA2), offset 0x124

The **CANNWDA1** and **CANNWDA2** registers hold the NEWDAT bits of the 32 message objects. By reading these bits, the CPU can check which message object has its data portion updated. The NEWDAT bit of a specific message object can be changed by three sources: (1) the CPU via the **CANIFnMCTL** register, (2) the message handler state machine after the reception of a data frame, or (3) the message handler state machine after a successful transmission.

The **CANNWDA1** register contains the NEWDAT bits of the first 16 message objects in the message RAM; the **CANNWDA2** register contains the NEWDAT bits of the second 16 message objects.

CAN New Data 1 (CANNWDA1)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x120
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	NEWDAT	RO	0x0000	New Data Bits 0: No new data has been written into the data portion of the corresponding message object by the message handler since the last time this flag was cleared by the CPU. 1: The message handler or the CPU has written new data into the data portion of the corresponding message object.

Register 34: CAN Message 1 Interrupt Pending (CANMSG1INT), offset 0x140

Register 35: CAN Message 2 Interrupt Pending (CANMSG2INT), offset 0x144

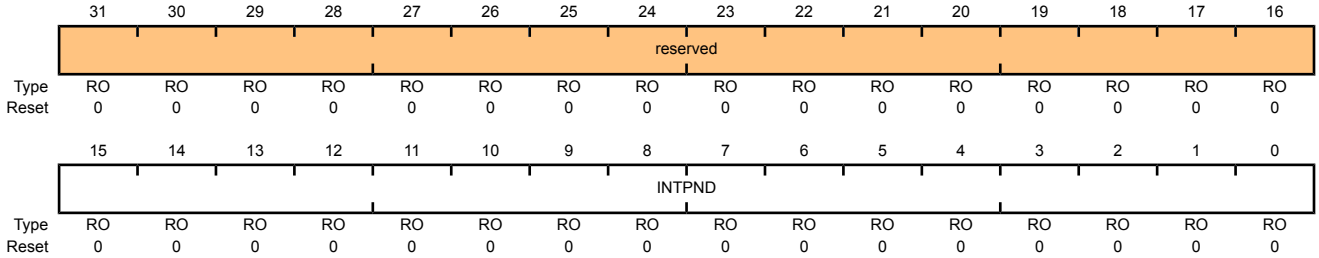
The **CANMSG1INT** and **CANMSG2INT** registers hold the INTPND bits of the 32 message objects. By reading these bits, the CPU can check which message object has an interrupt pending. The INTPND bit of a specific message object can be changed through two sources: (1) the CPU via the **CANIFnMCTL** register, or (2) the message handler state machine after the reception or transmission of a frame.

This field is also encoded in the **CANINT** register.

The **CANMSG1INT** register contains the INTPND bits of the first 16 message objects in the message RAM; the **CANMSG2INT** register contains the INTPND bits of the second 16 message objects.

CAN Message 1 Interrupt Pending (CANMSG1INT)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x140
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	INTPND	RO	0x0000	Interrupt Pending Bits 0: The corresponding message object is not the source of an interrupt. 1: The corresponding message object is the source of an interrupt.

Register 36: CAN Message 1 Valid (CANMSG1VAL), offset 0x160

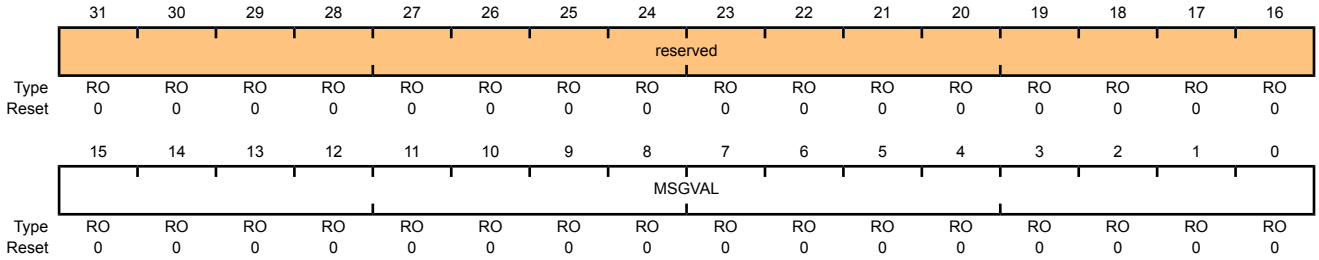
Register 37: CAN Message 2 Valid (CANMSG2VAL), offset 0x164

The **CANMSG1VAL** and **CANMSG2VAL** registers hold the MSGVAL bits of the 32 message objects. By reading these bits, the CPU can check which message object is valid. The message value of a specific message object can be changed with the **CANIFnMCTL** register.

The **CANMSG1VAL** register contains the MSGVAL bits of the first 16 message objects in the message RAM; the **CANMSG2VAL** register contains the MSGVAL bits of the second 16 message objects in the message RAM.

CAN Message 1 Valid (CANMSG1VAL)

CAN0 base: 0x4004.0000
CAN1 base: 0x4004.1000
Offset 0x160
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x0000	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MSGVAL	RO	0x0000	Message Valid Bits 0: The corresponding message object is not configured and is ignored by the message handler. 1: The corresponding message object is configured and should be considered by the message handler.

18 Universal Serial Bus (USB) Controller

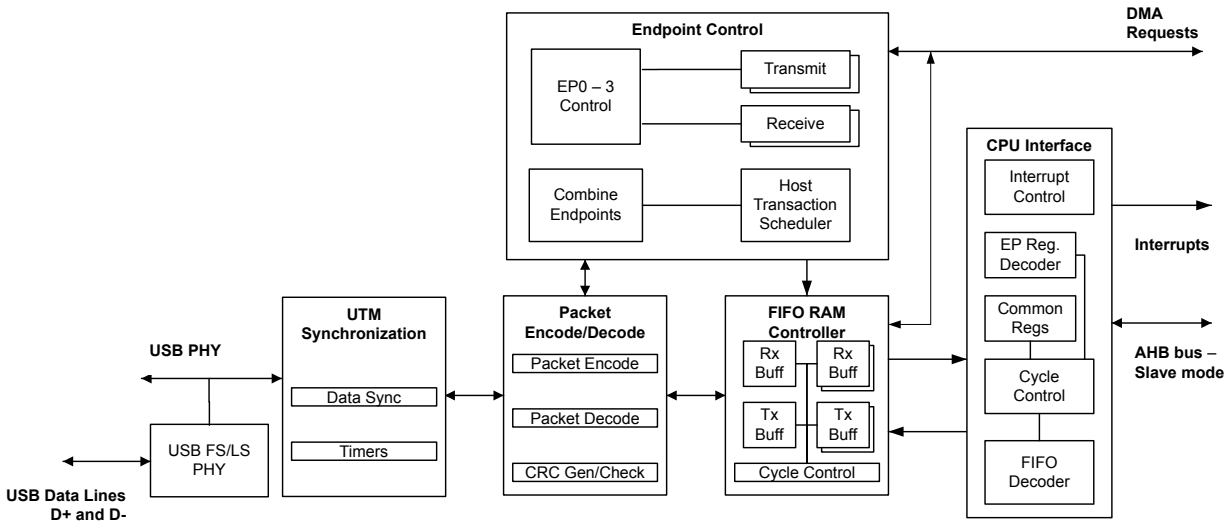
The Stellaris® USB controller operates as as a full-speed or low-speed function controller during point-to-point or multipoint (hub) communications with USB functions. The controller complies with the USB 2.0 standard, which includes suspend and resume signaling. Eight endpoints including two hard-wired for control transfers (one endpoint for IN and one endpoint for OUT) plus six endpoints defined by firmware along with a dynamic sizable FIFO support multiple packet queueing. DMA access to the FIFO allows minimal interference from system software. The controller has the capability to access an external power regulator through a power enable pad output (USB0EPEN) and power fault detect pad input (USB0PFLT).

The Stellaris® USB module has the following features:

- Standards-based
- USB 2.0 full-speed (12 Mbps) and low-speed (1.5 Mbps) operation
- USB Host mode
- Integrated PHY
- 4 transfer types: Control, Interrupt, Bulk, and Isochronous
- 8 endpoints
  - 1 dedicated control IN endpoint and 1 dedicated control OUT endpoint
  - 3 configurable IN endpoints and 3 configurable OUT endpoints
- 4 KB dedicated endpoint memory
  - Direct memory access (DMA)
  - One endpoint may be defined for double-buffered 1023-byte isochronous packet size

18.1 Block Diagram

Figure 18-1. USB Module Block Diagram



18.2 Functional Description

Note: A 9.1-kΩ resistor should be connected between the USB0RBIAS and ground. The 9.1-kΩ resistor should have a 1% tolerance and should be located in close proximity to the USB0RBIAS pin. Power dissipation in the resistor is low, so a chip resistor of any geometry may be used.

The Stellaris® USB controller provides the ability for the controller to switch from host controller to device controller functionality. The USB controller requires both A and B connectors in the system to provide host or device connectivity. If both connectors are present, the controller provides external signals to enable or disable power to the USB0VBUS pin on the USB connector when not in use. The controller can only be used in host or device mode and cannot be used in both modes simultaneously. However, the controller can be manually switched at run time if the system requires both host and device functionality.

18.2.1 Operation as a Device

This section describes the Stellaris® USB controller's actions when it is being used as a USB device. IN endpoints, OUT endpoints, entry into and exit from Suspend mode, and recognition of Start of Frame (SOF) are all described.

When in device mode, IN transactions are controlled by an endpoint's transmit interface and use the transmit endpoint registers for the given endpoint. OUT transactions are handled with an endpoint's receive interface and use the receive endpoint registers for the given endpoint.

When configuring the size of the FIFOs for endpoints, take into account the maximum packet size for an endpoint.

- **Bulk.** Bulk endpoints should be sized to be multiples of the maximum packet size (up to 64 bytes). For instance, if maximum packet size is 64 bytes, the FIFO should be configured to a multiple of 64-byte packets (64, 128, 192, or 256 bytes). This allows for efficient use of double buffering or packet splitting (described further in the following sections).

- **Interrupt.** Interrupt endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used.
- **Isochronous.** Isochronous endpoints are more flexible and can be up to 1023 bytes.
- **Control.** It is also possible to specify a separate control endpoint for a USB device. However, in most cases the USB device should use the dedicated control endpoint on the USB controller's endpoint 0.

18.2.1.1 Endpoints

When operating as a device, there are two dedicated control endpoints (IN and OUT) and six configurable endpoints (3 IN and 3 OUT) that can be used for communications with a host controller. The endpoint number associated with an endpoint is directly related to its register designation. For example, when the host is communicating with endpoint 1, all events will occur in the endpoint 1 register interface.

Endpoint 0 is a dedicated control endpoint used for all control transactions to endpoint 0 during enumeration or when any other control requests are made to endpoint 0. Endpoint 0 uses the first 64 bytes of the USB controller's FIFO RAM as a shared memory for both IN and OUT transactions.

The remaining six endpoints can be configured as control, bulk, interrupt, or isochronous endpoints. They should be treated as three configurable IN and three configurable OUT endpoints. The three endpoint pairs (endpoint 1, 2, and 3) are not required to have the same type for their IN and OUT endpoint configuration. For example, the OUT portion of an endpoint pair could be a bulk endpoint, while the IN portion of that endpoint pair could be an interrupt endpoint. The address and size of the FIFOs attached to each endpoint can be modified to fit the application's needs.

18.2.1.2 IN Transactions

When operating as a USB device, data for IN transactions is handled through the FIFOs attached to the transmit endpoints. The sizes of the FIFOs for the three configurable IN endpoints are determined by the **USBTXFIFOADD** register. The maximum size of a data packet that may be placed in a transmit endpoint's FIFO for transmission is programmable and is determined by the value written to the **USBTXMAXPn** register for that endpoint. The endpoint's FIFO can also be configured to use double-packet or single-packet buffering. When double-packet buffering is enabled, two data packets can be buffered in the FIFO, which also requires that the FIFO is at least two packets in size. When double-packet buffering is disabled, only one packet can be buffered, even if the packet size is less than half the FIFO size. The USB controller also supports a special mode for bulk endpoints that allows automatic splitting of a larger FIFO into multiple packets that are maximum packet size transfers.

Note: The maximum packet size set for any endpoint must not exceed the FIFO size. The **USBTXMAXPn** register should not be written to while there is data in the FIFO as unexpected results may occur.

Single-Packet Buffering

If the size of the transmit endpoint's FIFO is less than twice the maximum packet size for this endpoint (as set in the **USBTXFIFOSZ** register), only one packet can be buffered in the FIFO and single-packet buffering is required. When each packet is completely loaded into the transmit FIFO, the TXRDY bit in the **USBTXCSRLn** register needs to be set. If the AUTOSET bit in the **USBTXCSRHn** register is set, the TXRDY bit is automatically set when a maximum sized packet is loaded into the FIFO. For packet sizes less than the maximum, the TXRDY bit must be set manually. When the TXRDY bit is set, either manually or automatically, the packet is ready to be sent. When the packet has been

successfully sent, both TXRDY and FIFONE are cleared and the appropriate transmit endpoint interrupt signaled. At this point, the next packet can be loaded into the FIFO.

Double-Packet Buffering

If the size of the transmit endpoint's FIFO is at least twice the maximum packet size for this endpoint, two packets can be buffered in the FIFO and double-packet buffering is allowed. As each packet is loaded into the transmit FIFO, the TXRDY bit in the **USBTXCSRLn** register needs to be set. If the AUTOSET bit in the **USBTXCSRHn** register is set, the TXRDY bit is automatically set when a maximum sized packet is loaded into the FIFO. For packet sizes less than the maximum, TXRDY must be set manually. When the TXRDY bit is set, either manually or automatically, the packet is ready to be sent. After the first packet is loaded, TXRDY is immediately cleared and an interrupt is generated. A second packet can now be loaded into the transmit FIFO and TXRDY set again (either manually or automatically if the packet is the maximum size). At this point, both packets are ready to be sent. After each packet has been successfully sent, TXRDY is cleared and the appropriate transmit endpoint interrupt signaled to indicate that another packet can now be loaded into the transmit FIFO. The state of the FIFONE bit at this point indicates how many packets may be loaded. If the FIFONE bit is set, then there is another packet in the FIFO and only one more packet can be loaded. If the FIFONE bit is clear, then there are no packets in the FIFO and two more packets can be loaded.

Note: Double-packet buffering is disabled if an endpoint's corresponding EPn bit is set in the **USBTXDPKTBUFDIS** register. This bit is set by default, so it must be cleared to enable double-packet buffering.

Special Bulk Handling

The packets transferred in bulk operations are defined by the USB specification to be 8, 16, 32 or 64 bytes in size. For some system designs, however, it may be more convenient for the application software to write larger amounts of data to an endpoint in a single operation than can be transferred in a single USB operation.

To simplify this case, the Stellaris® USB controller includes a packet-splitting feature that allows larger data packets to be written to bulk transmit endpoints, which are then split into packets of an appropriate size for transfer across the USB bus. With this option, the **USBTXMAXPn** register uses the bottom 11 bits to define the payload for each individual transfer, while the top 5 bits define a multiplier. The application software can then write data packets of size multiplier × payload to the FIFO, which the USB controller then splits into individual packets of the stated payload for transmission over the USB bus. From the application software's point-of-view, the resulting operation does not differ from the transmission of a single USB packet except in the size of the packet written.

Note: Packet-splitting can only be used with bulk endpoints and, in accordance with the USB specification, the payload must be 8, 16, 32, or 64. The payload recorded in the **USBTXMAXPn** register must also match the wMaxPacketSize field of the Standard Endpoint Descriptor for the endpoint (see chapter 9 of the USB specification). The associated FIFO must also be large enough to accommodate the data packet prior to being split.

18.2.1.3 OUT Transactions as a Device

When in device mode, OUT transactions are handled through the USB controller receive FIFOs. The sizes of the receive FIFOs for the three configurable OUT endpoints are determined by the **USBRXFIFOADD** register. The maximum amount of data received by an endpoint in any packet is determined by the value written to the **USBRXMAXPn** register for that endpoint. When double-packet buffering is enabled, two data packets can be buffered in the FIFO. When double-packet buffering is disabled, only one packet can be buffered even if the packet is less than half the FIFO size. The

Stellaris® USB controller also supports a special mode for bulk endpoints that allows automatic splitting of a larger FIFO into multiple maximum packet size transfers.

Note: In all cases, the maximum packet size must not exceed the FIFO size.

Single-Packet Buffering

If the size of the receive endpoint FIFO is less than twice the maximum packet size for an endpoint, only one data packet can be buffered in the FIFO and single-packet buffering is required. When a packet is received and placed in the receive FIFO, the RXRDY and FULL bits in the **USBRXCSRLn** register are set and the appropriate receive endpoint is signaled, indicating that a packet can now be unloaded from the FIFO. After the packet has been unloaded, the RXRDY bit needs to be cleared in order to allow further packets to be received. This action also generates the acknowledge signaling to the host controller. If the AUTOCL bit in the **USBRXCSRHn** register is set and a maximum-sized packet is unloaded from the FIFO, the RXRDY and FULL bits are cleared automatically. For packet sizes less than the maximum, RXRDY must be cleared manually.

Double-Packet Buffering

If the size of the receive endpoint FIFO is at least twice the maximum packet size for the endpoint, two data packets can be buffered and double-packet buffering can be used. When the first packet is received and loaded into the receive FIFO, the RXRDY bit in the **USBRXCSRLn** register is set and the appropriate receive endpoint interrupt is signaled to indicate that a packet can now be unloaded from the FIFO.

Note: The FULL bit in **USBRXCSRLn** is not set when the first packet is received. It is only set if a second packet is received and loaded into the receive FIFO.

After each packet has been unloaded, the RXRDY bit needs to be cleared in order to allow further packets to be received. If the AUTOCL bit in the **USBRXCSRHn** register is set and a maximum-sized packet is unloaded from the FIFO, the RXRDY bit is cleared automatically. For packet sizes less than the maximum, RXRDY must be cleared manually. If the FULL bit was set when RXRDY is cleared, the USB controller first clears the FULL bit. It then sets RXRDY again to indicate that there is another packet waiting in the FIFO to be unloaded.

Note: Double-packet buffering is disabled if an endpoint's corresponding EPn bit is set in the **USBRXDPKTBUFDIS** register. This bit is set by default, so it must be cleared to enable double-packet buffering.

Special Bulk Handling

The packets transferred in bulk operations are defined by the USB specification to be 8, 16, 32, or 64 bytes in size. For some system designs, however, it may be more convenient for the application software to read larger amounts of data from an endpoint in a single operation than can be transferred in a single USB operation.

To simplify this case, the Stellaris® USB controller includes a packet-combining feature that combines the packets received across the USB bus into larger data packets prior to being read by the application software. With this option, the **USBRXMAXPn** register uses the bottom 11 bits to define the payload for each individual transfer, while the top 5 bits define a multiplier. The USB controller then combines the appropriate number of USB packets it receives into a single data packet of size multiplier × payload within the FIFO before asserting RXRDY to alert the application software that a packet in the FIFO is ready to be read. The size of the resulting packet is reported in the **USBRXCOUNTn** register. From the application software's point-of-view, the resulting operation does not differ from the receipt of a single USB packet except in the size of the packet read.

Note: Packet-combining can only be used with bulk endpoints. The payload recorded in the **USBRXMAXPn** register must also match the `wMaxPacketSize` field of the Standard Endpoint Descriptor for the endpoint (see chapter 9 of the USB specification). The associated FIFO must also be large enough to accommodate the combined data packet.

The `RXRDY` bit is only set when either the specified number of packets have been received or a "short" USB packet is received (that is, a packet of less than the specified payload for the endpoint). If a protocol is being used in which the endpoint receives bulk transfers that are a multiple of the recorded payload size with no short packet to terminate it, the **USBRXMAXPn** register should not be programmed to expect more packets than there are in the transfer (otherwise, the software will not be interrupted at the end of the transfer).

18.2.1.4 Scheduling

The device has no control over the scheduling of transactions as this is determined by the host controller. The Stellaris® USB controller can set up a transaction at any time. The USB controller will wait for the request from the host controller and generate an interrupt when the transaction is complete or if it was terminated due to some error. If the host controller makes a request and the device controller is not ready, the USB controller sends a busy response (NAK) to all requests until it is ready.

18.2.1.5 Additional Actions

The USB controller responds automatically to certain conditions on the USB bus or actions by the host controller: when the USB controller automatically stalls a control transfer and unexpected zero length OUT data packets.

Stalled Control Transfer

The USB controller automatically issues a STALL handshake to a control transfer under the following conditions:

1. The host sends more data during an OUT data phase of a control transfer than was specified in the device request during the SETUP phase. This condition is detected by the USB controller when the host sends an OUT token (instead of an IN token) after the last OUT packet has been unloaded and the `DATAEND` bit in the **USBCSRL0** register has been set.

2. The host requests more data during an IN data phase of a control transfer than was specified in the device request during the SETUP phase. This condition is detected by the USB controller when the host sends an IN token (instead of an OUT token) after the CPU has cleared `TXRDY` and set `DATAEND` in response to the ACK issued by the host to what should have been the last packet.

3. The host sends more than **USBRXMAXPn** bytes of data with an OUT data token.

4. The host sends more than a zero length data packet for the OUT status phase.

Zero Length OUT Data Packets

A zero-length OUT data packet is used to indicate the end of a control transfer. In normal operation, such packets should only be received after the entire length of the device request has been transferred.

However, if the host sends a zero-length OUT data packet before the entire length of device request has been transferred, it is signaling the premature end of the transfer. In this case, the USB controller

automatically flushes any IN token ready for the data phase from the FIFO and sets the SETUP bit in the **USBCSRL0** register.

18.2.1.6 Device Mode Suspend

When no activity has occurred on the USB bus for 3 ms, the USB controller automatically enters Suspend mode. If the Suspend interrupt has been enabled, an interrupt is generated at this time. When in Suspend mode, the PHY also goes into Suspend mode. When Resume signaling is detected, the USB controller exits Suspend mode and takes the PHY out of Suspend. If the Resume interrupt is enabled, an interrupt is generated. The USB controller can also be forced to exit Suspend mode by setting the RESUME bit in the **USBPOWER** register. When this bit is set, the USB controller exits Suspend mode and drives Resume signaling onto the bus. The RESUME bit is cleared after 10 ms (a maximum of 15 ms) to end Resume signaling.

To meet USB power requirements, the controller can be put into Deep Sleep. This keeps the controller in a static state. The USB controller is not able to Hibernate since this will cause all the internal states to be lost.

18.2.1.7 Start-of-Frame

When the USB controller is operating in device mode, it receives a Start-Of-Frame packet from the host once every millisecond. When the SOF packet is received, the 11-bit frame number contained in the packet is written into the **USBFRAME** register and an SOF interrupt is also signaled and can be handled by the application. Once the USB controller has started to receive SOF packets, it expects one every millisecond. If no SOF packet is received after 1.00358 ms, it is assumed that the packet has been lost and the **USBFRAME** register is not updated. The USB controller continues and resynchronizes these pulses to the received SOF packets when these packets are successfully received again.

18.2.1.8 USB Reset

When the USB controller is in device mode and a reset condition is detected on the USB bus, the USB controller automatically performs the following actions:

- Clears the **USBFADDR** register.
- Clears the **USBEPIDX** register.
- Flushes all endpoint FIFOs.
- Clears all control/status registers.
- Enables all endpoint interrupts.
- Generates a reset interrupt.

When the application software driving the USB controller receives a reset interrupt, it closes any open pipes and waits for bus enumeration to begin.

18.2.1.9 Connect/Disconnect

The USB controller connection to the USB bus is controlled by software. The USB PHY can be switched between normal mode and non-driving mode by setting or clearing the SOFTCONN bit of the **USBPOWER** register. When this SOFTCONN bit is set, the PHY is placed in its normal mode and the USB0DP/USB0DM lines of the USB bus are enabled. At the same time, the USB controller is placed into a state, in which it will not respond to any USB signaling except a USB reset.

When the SOFTCONN bit is cleared, the PHY is put into non-driving mode, USB0DP and USB0DM are tristated, and the USB controller appears to other devices on the USB bus as if it has been disconnected. This is the default so the USB controller appears disconnected until the SOFTCONN bit has been set. The application software can then choose when to set the PHY into its normal mode. Systems with a lengthy initialization procedure may use this to ensure that initialization is complete and the system is ready to perform enumeration before connecting to the USB. Once the SOFTCONN bit has been set, the USB controller can be disconnected by clearing this bit.

Note: The USB controller does not generate an interrupt when the device is connected to the host. However, an interrupt is generated when the host terminates a session.

18.2.2 Operation as a Host

When the Stellaris® USB controller is operating in host mode, it can either be used for point-to-point communications with another USB device or, when attached to a hub, for communication with multiple devices. Full-speed and low-speed USB devices are supported, both for point-to-point communication and for operation through a hub. The USB controller automatically carries out the necessary transaction translation needed to allow a low-speed or full-speed device to be used with a USB 2.0 hub. Control, bulk, isochronous, and interrupt transactions are supported. This section describes the USB controller's actions when it is being used as a USB host. Configuration of IN endpoints, OUT endpoints, entry into and exit from Suspend mode, and reset are all described.

When in host mode, IN transactions are controlled by an endpoint's receive interface. All IN transactions use the receive endpoint registers and all OUT endpoints use the transmit endpoint registers for a given endpoint. As in device mode, the FIFOs for endpoints should take into account the maximum packet size for an endpoint.

- **Bulk.** Bulk endpoints should be sized to be multiples of the maximum packet size (up to 64 bytes). For instance, if maximum packet size is 64 bytes, the FIFO should be configured to a multiple of 64-byte packets (64, 128, 192, or 256 bytes). This allows for efficient use of double buffering or packet splitting (described further in the following sections).
- **Interrupt.** Interrupt endpoints should be the size of the maximum packet (up to 64 bytes) or twice the maximum packet size if double buffering is used.
- **Isochronous.** Isochronous endpoints are more flexible and can be up to 1023 bytes.
- **Control.** It is also possible to specify a separate control endpoint to communicate with a device. However, in most cases the USB controller should use the dedicated control endpoint to communicate with a device's endpoint 0.

18.2.2.1 Endpoints

The endpoint registers are used to control the USB endpoint interfaces used to communicate with device(s) that are connected. There is a dedicated control IN endpoint, a dedicated control OUT endpoint, three configurable OUT endpoints, and three configurable IN endpoints.

The dedicated control interface can only be used for control transactions to endpoint 0 of devices. These control transactions are used during enumeration or other control functions that communicate using endpoint 0 of devices. This control endpoint shares the first 64 bytes of the USB controller's FIFO RAM for IN and OUT transactions. The remaining IN and OUT interfaces can be configured to communicate with control, bulk, interrupt, or isochronous device endpoints.

These USB interfaces can be used to simultaneously schedule as many as three independent OUT and three independent IN transactions to any endpoints on any device. The IN and OUT controls are paired in three sets of registers. However, they can be configured to communicate with different

types of endpoints and different endpoints on devices. For example, the first pair of endpoint controls can be split so that the OUT portion is communicating with a device's bulk OUT endpoint 1, while the IN portion is communicating with a device's interrupt IN endpoint 2.

Before accessing any device, whether for point-to-point communications or for communications via a hub, the relevant **USBRXFUNCADDRn** or **USBTXFUNCADDRn** registers need to be set for each receive or transmit endpoint to record the address of the device being accessed.

The USB controller also supports connections to devices through a USB hub by providing a register that specifies the hub address and port of each USB transfer. The FIFO address and size are customizable and can be specified for each USB IN and OUT transfer. This includes allowing one FIFO per transaction, sharing a FIFO across transactions, and allowing for double-buffered FIFOs.

18.2.2.2 IN Transactions as a Host

IN transactions are handled in a similar manner to the way in which OUT transactions are handled when the USB controller is in Device mode except that the transaction first needs to be initiated by setting the REQPKT bit in **USBCSRL0**. This indicates to the transaction scheduler that there is an active transaction on this endpoint. The transaction scheduler then sends an IN token to the target device. When the packet is received and placed in the receive FIFO, the RXRDY bit in **USBCSRL0** is set and the appropriate receive endpoint interrupt is signaled to indicate that a packet can now be unloaded from the FIFO.

When the packet has been unloaded, RXRDY should be cleared. The AUTOCL bit in the **USBRXCSRHn** register can be used to have RXRDY automatically cleared when a maximum-sized packet has been unloaded from the FIFO. There is also an AUTORQ bit in **USBRXCSRHn** which causes the REQPKT bit to be automatically set when the RXRDY bit is cleared. The AUTOCL and AUTORQ bits can be used with DMA accesses to perform complete bulk transfers without main processor intervention. When the RXRDY bit is cleared, the controller will send an acknowledge to the device. When there is a known number of packets to be transferred, the **USBRQPKTCOUNTn** register associated with the endpoint should be set to the number of packets to be transferred. The USB controller decrements the value in the **USBRQPKTCOUNTn** register following each request. When the **USBRQPKTCOUNTn** value decrements to 0, the AUTORQ bit is cleared to prevent any further transactions being attempted. For cases where the size of the transfer is unknown, **USBRQPKTCOUNTn** should be left set to zero. AUTORQ then remains set until cleared by the reception of a short packet (that is, less than MaxP) such as may occur at the end of a bulk transfer.

If the device responds to a bulk or interrupt IN token with a NAK, the USB host controller keeps retrying the transaction until any NAK Limit that has been set has been reached. If the target device responds with a STALL, however, the USB host controller does not retry the transaction but interrupts the CPU with the STALLED bit in the **USBCSRL0** register set. If the target device does not respond to the IN token within the required time, or there was a CRC or bit-stuff error in the packet, the USB host controller retries the transaction. If after three attempts the target device has still not responded, the USB host controller clears the REQPKT bit and interrupts the CPU by setting the ERROR bit in the **USBCSRL0** register.

18.2.2.3 Out Transactions as a Host

OUT transactions are handled in a similar manner to the way in which IN transactions are handled when the USB controller is in Device mode. The TXRDY bit in the **USBTXCSRLn** register needs to be set as each packet is loaded into the transmit FIFO. Again, setting the AUTOSET bit in the **USBTXCSRHn** register automatically sets TXRDY when a maximum-sized packet has been loaded into the FIFO. Furthermore, AUTOSET can be used with a DMA controller to perform complete bulk transfers without software intervention.

If the target device responds to the OUT token with a NAK, the USB host controller keeps retrying the transaction until the NAK Limit that has been set has been reached. However, if the target device responds with a STALL, the USB controller does not retry the transaction but interrupts the main processor by setting the STALLED bit in the **USBTXCSRLn** register. If the target device does not respond to the OUT token within the required time, or there was a CRC or bit-stuff error in the packet, the USB host controller retries the transaction. If after three attempts the target device has still not responded, the USB controller flushes the FIFO and interrupts the main processor by setting the ERROR bit in the **USBTXCSRLn** register.

18.2.2.4 Transaction Scheduling

Scheduling of transactions is handled automatically by the USB host controller. The host controller allows configuration of the endpoint communication scheduling based on the type of endpoint transaction. Interrupt transactions can be scheduled to occur in the range of every frame to every 255 frames in 1 frame increments. Bulk endpoints do not allow scheduling parameters, but do allow for a NAK timeout in the event an endpoint on a device is not responding. Isochronous endpoints can be scheduled from every frame to every 2^{16} frames, in powers of 2.

The USB controller maintains a frame counter. If the target device is a full-speed device, the USB controller automatically sends an SOF packet at the start of each frame and increments the frame counter. If the target device is a low-speed device, a ‘K’ state is transmitted on the bus to act as a “keep-alive” to stop the low-speed device from going into Suspend mode.

After the SOF packet has been transmitted, the USB host controller cycles through all the configured endpoints looking for active transactions. An active transaction is defined as a receive endpoint for which the REQPKT bit is set or a transmit endpoint for which the TXRDY bit and/or the FIFONE bit is set.

An active isochronous or interrupt transaction starts only if it is found on the first transaction scheduler cycle of a frame and if the interval counter for that endpoint has counted down to zero. This ensures that only one interrupt or isochronous transaction occurs per endpoint every n frames, where n is the interval set via the **USBTXINTERVALn** or **USBRXINTERVALn** register for that endpoint.

An active bulk transaction starts immediately, provided there is sufficient time left in the frame to complete the transaction before the next SOF packet is due. If the transaction needs to be retried (for example, because a NAK was received or the target device did not respond), then the transaction is not retried until the transaction scheduler has first checked all the other endpoints for active transactions. This ensures that an endpoint that is sending a lot of NAKs does not block other transactions on the bus. The core also allows the user to specify a limit to the length of time for NAKs to be received from a target device before the endpoint times out.

18.2.2.5 USB Hubs

The following setup requirements apply to the USB host controller only if it is used with a USB hub. When a full- or low-speed device is connected to the USB controller via a USB 2.0 hub, details of the hub address and the hub port also need to be recorded in the corresponding **USBRXHUBADDRn** and **USBRXHUBPORTn** or the **USBTXHUBADDRn** and **USBTXHUBPORTn** registers. In addition, the speed at which the device operates (full or low) needs to be recorded in the **USBTYPE0** (endpoint 0), **USBTXTYPEn**, or **USBRXTYPEn** registers for each endpoint that is accessed by the device.

For hub communications, the settings in these registers record the current allocation of the endpoints to the attached USB devices. To maximize the number of devices supported, the USB host controller allows this allocation to be changed dynamically by simply updating the address and speed information recorded in these registers. Any changes in the allocation of endpoints to device functions need to be made following the completion of any on-going transactions on the endpoints affected.

18.2.2.6 Babble

The USB host controller does not start a transaction until the bus has been inactive for at least the minimum inter-packet delay. It also does not start a transaction unless it can be finished before the end of the frame. If the bus is still active at the end of a frame, then the USB host controller assumes that the target device to which it is connected has malfunctioned and the USB controller suspends all transactions and generates a babble interrupt.

18.2.2.7 Host Suspend

If the SUSPEND bit in the **USBPOWER** register is set, the USB host controller completes the current transaction then stops the transaction scheduler and frame counter. No further transactions are started and no SOF packets are generated.

To exit Suspend mode, the RESUME bit is set and the SUSPEND bit is cleared. While the RESUME bit is High, the USB host controller generates Resume signaling on the bus. After 20 ms, the RESUME bit should be cleared, at which point the frame counter and transaction scheduler start. The host supports the detection of a remote wake-up.

18.2.2.8 USB Reset

If the RESET bit in the **USBPOWER** register is set, the USB host controller generates USB Reset signaling on the bus. The RESET bit should be set for at least 20 ms to ensure correct resetting of the target device. After the CPU has cleared the bit, the USB host controller starts its frame counter and transaction scheduler.

18.2.2.9 Connect/Disconnect

A session is started by setting the SESSION bit in the **USBDEVCTL** register. This enables the USB controller to wait for a device to be connected. When a device is detected, a connect interrupt is generated. The speed of the device that has been connected can be determined by reading the **USBDEVCTL** register where the FSDEV bit is High for a full-speed device and the LSDEV bit is High for a low-speed device. The USB controller should generate a reset to the device and then the USB host controller can begin device enumeration. If the device is disconnected while a session is in progress, a disconnect interrupt is generated.

18.2.3 DMA Operation

The USB peripheral provides an interface connected to the µDMA controller. The DMA operation of the USB is enabled through the **USBTXCSRHn** and **USBRXCSRHn** registers, for the TX and RX channels respectively. When DMA operation is enabled, the USB asserts a DMA request on the enabled receive or transmit channel when the associated FIFO can transfer data. When either FIFO can transfer data, the burst request for that channel is asserted. Since only burst request is used, the arbitration size for the µDMA channel must be set to a size that divides into the FIFO size. For example, if the FIFO is 64 bytes, then the arbitration size could be a power of 2 up to 64.

If DMA is enabled, then the µDMA controller triggers an interrupt when a transfer is complete. The interrupt occurs on the USB interrupt vector. Therefore, if interrupts are used for USB operation and DMA is enabled, the USB interrupt handler must be designed to handle the µDMA completion interrupt.

Care must be taken when using a DMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of the RxMaxP bit in the **USBRXCSRHn** register. The RXRDY bit is cleared as follows.

Table 18-1. Remainder (RxMaxP/4)

Value	Description
0	RxMaxP = 64 bytes
1	RxMaxP = 61 bytes
2	RxMaxP = 62 bytes
3	RXMaxP = 63 bytes

Table 18-2. Actual Bytes Read

Value	Description
0	RxMaxP
1	RxMaxP+3
2	RxMaxP+2
3	RxMaxP+1

Table 18-3. Packet Sizes That Will Clear RXRDY

Value	Description
0	RxMaxP, RxMaxP-1, RxMaxP-2, RxMaxP-3
1	RxMaxP
2	RxMaxP, RxMaxP-1
3	RxMaxP, RxMaxP-1, RxMaxP-2

To enable DMA operation for the endpoint receive channel, the DMAEN bit of the **USBRXCSRHn** register should be set. To enable DMA operation for the endpoint transmit channel, the DMAEN bit of the **USBTXCSRHn** register should be set.

See “Micro Direct Memory Access (µDMA)” on page 194 for more details about programming the µDMA controller.

18.3 Initialization and Configuration

The initial configuration in all cases requires that the processor enable the USB controller and USB controller’s physical layer interface (PHY) before setting any registers. The next step is to enable the USB PLL so that the correct clocking is provided to the PHY. To ensure that voltage is not supplied to the bus incorrectly, the external power control signal, USB0EPEN, should be de-asserted on start up. This requires setting the USB0EPEN and USB0PFLT pins to be controlled by the USB controller and not have their default GPIO behavior.

The USB controller provides a method to set the current operating mode of the USB controller. This register should be written with the desired default mode so that the controller can respond to external USB events.

18.3.1 Pin Configuration

When using the device controller portion of the USB controller in a system that also provides host functionality, the power to VBUS must be disabled to allow the external host controller to supply power. Usually, the USB0EPEN signal is used to control the external regulator and should be de-asserted to avoid having two devices driving the USB0VBUS power pin on the USB connector.

When the USB controller is acting as a host, it is in control of two signals that are attached to an external voltage supply that provides power to VBUS. The host controller uses the USB0EPEN signal

to enable or disable power to the USB0VBUS pin on the USB connector. There is also an input pin, USB0PFLT, which provides feedback when there has been a power fault on VBUS. The USB0PFLT signal can be configured to either automatically de-assert the USB0EPEN signal to disable power, and/or it can generate an interrupt to the main processor to allow it to handle the power fault condition. The polarity and actions related to both USB0EPEN and USB0PFLT are fully configurable in the USB controller. The controller also provides interrupts on device insertion and removal to allow the host controller code to respond to these external events.

18.3.2 Endpoint Configuration

In order to start communication on host or device mode, the endpoint registers must first be configured. In Host mode, this provides a connection between an endpoint register and an endpoint on a device. In Device mode, this provides the setup for a given endpoint before enumerating to the host controller.

In both cases, the endpoint 0 configuration is limited as this is a fixed function, fixed FIFO size endpoint. In Device and Host modes, the endpoint requires little setup but does require a software-based state machine to progress through the setup, data, and status phases of a standard control transaction. In Device mode, the configuration of the remaining endpoints is done once before enumerating and then only changed if an alternate configuration is selected by the host controller. In Host mode, the endpoints must be configured to operate as control, bulk, interrupt or isochronous mode. Once the type of endpoint is configured, a FIFO area must be assigned to each endpoint. In the case of bulk, control and interrupt endpoints, each has a maximum of 64 bytes per transaction. Isochronous endpoints can have packets with up to 1023 bytes per packet. In either mode, the maximum packet size for the given endpoint must be set prior to sending or receiving data.

Configuring each endpoint's FIFO involves reserving a portion of the overall USB FIFO RAM to each endpoint. The total FIFO RAM available is 4 Kbytes with the first 64 bytes in use by endpoint 0. The endpoint's FIFO does not have to be the same size as the maximum packet size in all cases as the controller can automatically split for bulk transactions if the FIFO is larger than the maximum packet size. The FIFO can also be configured as a double-buffered FIFO so that interrupts occur at the end of each packet and allow filling the other half of the FIFO.

If operating as a device, the USB device controllers' soft connect should be enabled when the device is ready to start communications. This indicates to the host controller that the device is ready to start the enumeration process. If operating as a host controller, the device soft connect should be disabled and power should be provided to VBUS via the USB0EPEN signal.

18.4 Register Map

Table 18-4 on page 573 lists the registers. All addresses given are relative to the USB base address of 0x4005.0000.

Table 18-4. Universal Serial Bus (USB) Controller Register Map

Offset	Name	Type	Reset	Description	See page
0x000	USBFADDR	R/W	0x00	USB Device Functional Address	577
0x001	USBPOWER	R/W	0x20	USB Power	578
0x002	USBTXIS	RO	0x0000	USB Transmit Interrupt Status	580
0x004	USBRXIS	RO	0x0000	USB Receive Interrupt Status	581

Offset	Name	Type	Reset	Description	See page
0x006	USBTXIE	R/W	0x000F	USB Transmit Interrupt Enable	582
0x008	USBRXIE	R/W	0x000E	USB Receive Interrupt Enable	583
0x00A	USBIS	RO	0x00	USB General Interrupt Status	584
0x00B	USBIE	R/W	0x06	USB Interrupt Enable	586
0x00C	USBFRAME	RO	0x0000	USB Frame Value	588
0x00F	USBTEST	R/W	0x00	USB Test Mode	590
0x020	USBFIFO0	R/W	0x0000.0000	USB FIFO Endpoint 0	592
0x024	USBFIFO1	R/W	0x0000.0000	USB FIFO Endpoint 1	592
0x028	USBFIFO2	R/W	0x0000.0000	USB FIFO Endpoint 2	592
0x02C	USBFIFO3	R/W	0x0000.0000	USB FIFO Endpoint 3	592
0x060	USBDEVCTL	R/W	0x80	USB Device Control	593
0x062	USBTXFIFOSZ	R/W	0x00	USB Transmit Dynamic FIFO Sizing	595
0x063	USBRXFIFOSZ	R/W	0x00	USB Receive Dynamic FIFO Sizing	595
0x064	USBTXFIFOADD	R/W	0x0000	USB Transmit FIFO Start Address	596
0x066	USBRXFIFOADD	R/W	0x0000	USB Receive FIFO Start Address	596
0x07A	USBCONTIM	R/W	0x5C	USB Connect Timing	597
0x07D	USBFSEOF	R/W	0x77	USB Full-Speed Last Transaction to End of Frame Timing	598
0x07E	USBLSEOF	R/W	0x72	USB Low-Speed Last Transaction to End of Frame Timing	599
0x080	USBTXFUNCADDR0	R/W	0x00	USB Transmit Functional Address Endpoint 0	600
0x082	USBTXHUBADDR0	R/W	0x00	USB Transmit Hub Address Endpoint 0	601
0x083	USBTXHUBPORT0	R/W	0x00	USB Transmit Hub Port Endpoint 0	602
0x088	USBTXFUNCADDR1	R/W	0x00	USB Transmit Functional Address Endpoint 1	600
0x08A	USBTXHUBADDR1	R/W	0x00	USB Transmit Hub Address Endpoint 1	601
0x08B	USBTXHUBPORT1	R/W	0x00	USB Transmit Hub Port Endpoint 1	602
0x08C	USBRXFUNCADDR1	R/W	0x00	USB Receive Functional Address Endpoint 1	603
0x08E	USBRXHUBADDR1	R/W	0x00	USB Receive Hub Address Endpoint 1	604
0x08F	USBRXHUBPORT1	R/W	0x00	USB Receive Hub Port Endpoint 1	605
0x090	USBTXFUNCADDR2	R/W	0x00	USB Transmit Functional Address Endpoint 2	600
0x092	USBTXHUBADDR2	R/W	0x00	USB Transmit Hub Address Endpoint 2	601
0x093	USBTXHUBPORT2	R/W	0x00	USB Transmit Hub Port Endpoint 2	602
0x094	USBRXFUNCADDR2	R/W	0x00	USB Receive Functional Address Endpoint 2	603
0x096	USBRXHUBADDR2	R/W	0x00	USB Receive Hub Address Endpoint 2	604

Offset	Name	Type	Reset	Description	See page
0x097	USBRXHUBPORT2	R/W	0x00	USB Receive Hub Port Endpoint 2	605
0x098	USBTXFUNCADDR3	R/W	0x00	USB Transmit Functional Address Endpoint 3	600
0x09A	USBTXHUBADDR3	R/W	0x00	USB Transmit Hub Address Endpoint 3	601
0x09B	USBTXHUBPORT3	R/W	0x00	USB Transmit Hub Port Endpoint 3	602
0x09C	USBRXFUNCADDR3	R/W	0x00	USB Receive Functional Address Endpoint 3	603
0x09E	USBRXHUBADDR3	R/W	0x00	USB Receive Hub Address Endpoint 3	604
0x09F	USBRXHUBPORT3	R/W	0x00	USB Receive Hub Port Endpoint 3	605
0x0E	USBEPIDX	R/W	0x0000	USB Endpoint Index	589
0x102	USBCSRL0	W1C	0x00	USB Control and Status Endpoint 0 Low	607
0x103	USBCSRH0	W1C	0x00	USB Control and Status Endpoint 0 High	610
0x108	USBCOUNT0	RO	0x00	USB Receive Byte Count Endpoint 0	612
0x10A	USBTYPE0	R/W	0x00	USB Type Endpoint 0	613
0x10B	USBNAKLMT	R/W	0x00	USB NAK Limit	614
0x110	USBTXMAXP1	R/W	0x0000	USB Maximum Transmit Data Endpoint 1	606
0x112	USBTXCSRL1	R/W	0x00	USB Transmit Control and Status Endpoint 1 Low	615
0x113	USBTXCSRH1	R/W	0x00	USB Transmit Control and Status Endpoint 1 High	618
0x114	USBRXMAXP1	R/W	0x0000	USB Maximum Receive Data Endpoint 1	621
0x116	USBRXCSRL1	R/W	0x00	USB Receive Control and Status Endpoint 1 Low	622
0x117	USBRXCSRH1	R/W	0x00	USB Receive Control and Status Endpoint 1 High	625
0x118	USBRXCOUNT1	RO	0x0000	USB Receive Byte Count Endpoint 1	628
0x11A	USBTXTYPE1	R/W	0x00	USB Host Transmit Configure Type Endpoint 1	629
0x11B	USBTXINTERVAL1	R/W	0x00	USB Host Transmit Interval Endpoint 1	631
0x11C	USBRXTYPE1	R/W	0x00	USB Host Configure Receive Type Endpoint 1	632
0x11D	USBRXINTERVAL1	R/W	0x00	USB Host Receive Polling Interval Endpoint 1	634
0x120	USBTXMAXP2	R/W	0x0000	USB Maximum Transmit Data Endpoint 2	606
0x122	USBTXCSRL2	R/W	0x00	USB Transmit Control and Status Endpoint 2 Low	615
0x123	USBTXCSRH2	R/W	0x00	USB Transmit Control and Status Endpoint 2 High	618
0x124	USBRXMAXP2	R/W	0x0000	USB Maximum Receive Data Endpoint 2	621
0x126	USBRXCSRL2	R/W	0x00	USB Receive Control and Status Endpoint 2 Low	622
0x127	USBRXCSRH2	R/W	0x00	USB Receive Control and Status Endpoint 2 High	625
0x128	USBRXCOUNT2	RO	0x0000	USB Receive Byte Count Endpoint 2	628
0x12A	USBTXTYPE2	R/W	0x00	USB Host Transmit Configure Type Endpoint 2	629
0x12B	USBTXINTERVAL2	R/W	0x00	USB Host Transmit Interval Endpoint 2	631

Offset	Name	Type	Reset	Description	See page
0x12C	USBRXTYPE2	R/W	0x00	USB Host Configure Receive Type Endpoint 2	632
0x12D	USBRXINTERVAL2	R/W	0x00	USB Host Receive Polling Interval Endpoint 2	634
0x130	USBTXMAXP3	R/W	0x0000	USB Maximum Transmit Data Endpoint 3	606
0x132	USBTXCSRL3	R/W	0x00	USB Transmit Control and Status Endpoint 3 Low	615
0x133	USBTXCSRH3	R/W	0x00	USB Transmit Control and Status Endpoint 3 High	618
0x134	USBRXMAXP3	R/W	0x0000	USB Maximum Receive Data Endpoint 3	621
0x136	USBRXCSRL3	R/W	0x00	USB Receive Control and Status Endpoint 3 Low	622
0x137	USBRXCSRH3	R/W	0x00	USB Receive Control and Status Endpoint 3 High	625
0x138	USBRXCOUNT3	RO	0x0000	USB Receive Byte Count Endpoint 3	628
0x13A	USBTXTYPE3	R/W	0x00	USB Host Transmit Configure Type Endpoint 3	629
0x13B	USBTXINTERVAL3	R/W	0x00	USB Host Transmit Interval Endpoint 3	631
0x13C	USBRXTYPE3	R/W	0x00	USB Host Configure Receive Type Endpoint 3	632
0x13D	USBRXINTERVAL3	R/W	0x00	USB Host Receive Polling Interval Endpoint 3	634
0x304	USBRQPKTCOUNT1	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 1	635
0x308	USBRQPKTCOUNT2	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 2	635
0x30C	USBRQPKTCOUNT3	R/W	0x0000	USB Request Packet Count in Block Transfer Endpoint 3	635
0x340	USBRXDPKTBUFDIS	R/W	0x0000	USB Receive Double Packet Buffer Disable	636
0x342	USBTXDPKTBUFDIS	R/W	0x0000	USB Transmit Double Packet Buffer Disable	637
0x400	USBEPC	R/W	0x0000.0000	USB External Power Control	638
0x404	USBEPCRIS	RO	0x0000.0000	USB External Power Control Raw Interrupt Status	641
0x408	USBEPCIM	R/W	0x0000.0000	USB External Power Control Interrupt Mask	642
0x40C	USBEPCISC	R/W	0x0000.0000	USB External Power Control Interrupt Status and Clear	643
0x410	USBDRRIS	RO	0x0000.0000	USB Device Resume Raw Interrupt Status	644
0x414	USBDRIM	R/W	0x0000.0000	USB Device Resume Interrupt Mask	645
0x418	USBDRISC	W1C	0x0000.0000	USB Device Resume Interrupt Status and Clear	646
0x41C	USBGPCS	R/W	0x0000.0000	USB General-Purpose Control and Status	647

18.5 Register Descriptions

The LM3S5749 USB controller is configured to the communication mode specified in the USB0 bit field in the **DC6** register (page 115):

- Host or device (USB0 set to 0x2)

Register 1: USB Device Functional Address (USBFADDR), offset 0x000

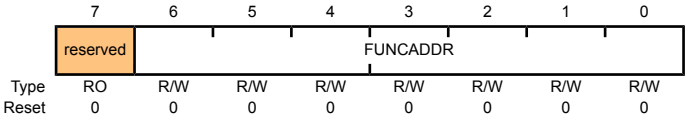
Device

USBFADDR is an 8-bit register that should be written with the 7-bit address of the device part of the transaction.

When the USB controller is being used in Device mode (HOST bit in **USBDEVCTL** register is 0), this register should be written with the address received through a SET_ADDRESS command, which is then used for decoding the function address in subsequent token packets.

USB Device Functional Address (USBFADDR)

Base 0x4005.0000
Offset 0x000
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	FUNCADDR	R/W	0x00	Function Address Function Address of Device as received through SET_ADDRESS.

Register 2: USB Power (USBPOWER), offset 0x001

Host

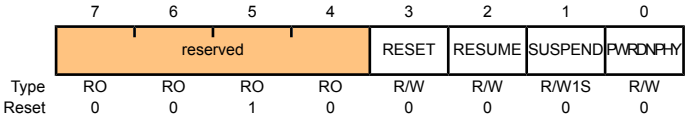
USBPOWER is an 8-bit register that is used for controlling Suspend and Resume signaling, and some basic operational aspects of the USB controller.

Device

Host Mode

USB Power (USBPOWER)

Base 0x4005.0000
Offset 0x001
Type R/W, reset 0x20

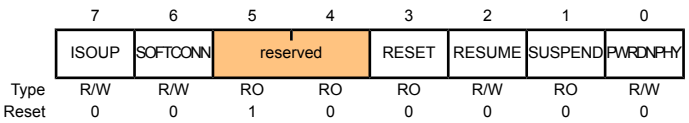


Bit/Field	Name	Type	Reset	Description
7:4	reserved	RO	0x02	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	RESET	R/W	0	Reset This bit is set to enable Reset signaling on the bus and cleared to end Reset signaling on the bus.
2	RESUME	R/W	0	Resume Signaling Set by the CPU to generate Resume signaling when the device is in Suspend mode. The CPU should clear this bit after 20 ms.
1	SUSPEND	R/W1S	0	Suspend Mode This bit is written to 1 by the CPU to enter Suspend mode. Writing a 0 does nothing.
0	PWRDNPHY	R/W	0	Power Down PHY Set by the CPU to power down the internal USB PHY.

Device Mode

USB Power (USBPOWER)

Base 0x4005.0000
Offset 0x001
Type R/W, reset 0x20



Bit/Field	Name	Type	Reset	Description
7	ISOUP	R/W	0	ISO Update When set by the CPU, the USB controller waits for an SOF token from the time TXRDY is set before sending the packet. If an IN token is received before an SOF token, then a zero-length data packet is sent. **Note:** Only valid for isochronous transfers.
6	SOFTCONN	R/W	0	Soft Connect/Disconnect The USB D+/D- lines are enabled when this bit is set by the CPU, and tri-stated when this bit is cleared by the CPU.
5:4	reserved	RO	0x2	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	RESET	RO	0	Reset This bit is set when Reset signaling is present on the bus.
2	RESUME	R/W	0	Resume Signaling Set by the CPU to generate Resume signaling when the device is in Suspend mode. The CPU should clear this bit after 10 ms (a maximum of 15 ms) to end Resume signaling.
1	SUSPEND	RO	0	Suspend Mode This bit is set on entry into Suspend mode. It is cleared when the CPU reads the interrupt register or sets the RESUME bit above.
0	PWRDNPHY	R/W	0	Power Down PHY Set by the CPU to power down the internal USB PHY.

Register 3: USB Transmit Interrupt Status (USBTXIS), offset 0x002

Host

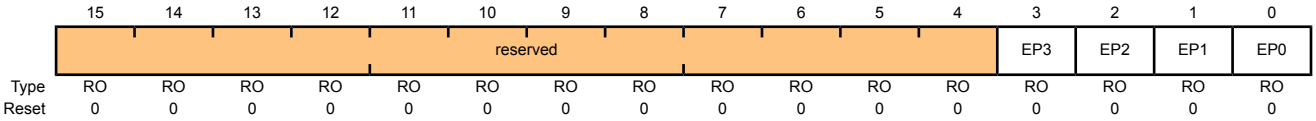
Device

USBTXIS is a 16-bit read-only register that indicates which interrupts are currently active for endpoint 0 and the transmit endpoints 1–3. The meaning of the EPn bits in this register are based on the mode of the device. For the EP1, EP2 and EP3 bits, these bits always indicate that the USB controller is sending data; however, in Host mode, these are the three configurable OUT endpoints; while in device mode, these are the three configurable IN endpoints. The EP0 bit is special in Host and Device modes and indicates that either a control IN or control OUT endpoint has generated an interrupt.

Note: Bits relating to endpoints that have not been configured always return 0. Note also that all active interrupts are cleared when this register is read.

USB Transmit Interrupt Status (USBTXIS)

Base 0x4005.0000
Offset 0x002
Type RO, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EP3	RO	0	TX Endpoint 3 Interrupt
2	EP2	RO	0	TX Endpoint 2 Interrupt
1	EP1	RO	0	TX Endpoint 1 Interrupt
0	EP0	RO	0	TX and RX Endpoint 0 Interrupt

Register 4: USB Receive Interrupt Status (USBRXIS), offset 0x004

Host

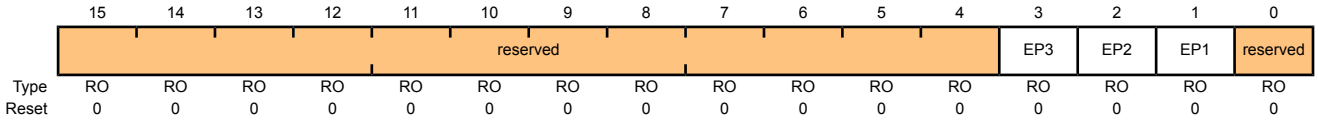
USBRXIS is a 16-bit read-only register that indicates which of the interrupts for receive endpoints 1–3 are currently active.

Device

Note: Bits relating to endpoints that have not been configured always return 0. Note also that all active interrupts are cleared when this register is read.

USB Receive Interrupt Status (USBRXIS)

Base 0x4005.0000
Offset 0x004
Type RO, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EP3	RO	0	RX Endpoint 3 Interrupt
2	EP2	RO	0	RX Endpoint 2 Interrupt
1	EP1	RO	0	RX Endpoint 1 Interrupt
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 5: USB Transmit Interrupt Enable (USBTXIE), offset 0x006

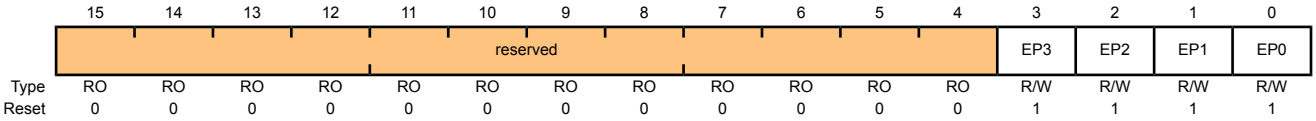
Host

Device

USBTXIE is a 16-bit register that provides interrupt enable bits for the interrupts in **USBTXIS**. When a bit in **USBTXIE** is set to 1, the USB interrupt to the processor is asserted when the corresponding interrupt bit in the **USBTXIS** register is set. When a bit is cleared to 0, the interrupt in **USBTXIS** is still set but the USB interrupt to the processor is not asserted. On reset, the bits corresponding to endpoint 0 and transmit endpoints 1-3 are set to 1, while the remaining bits are set to 0.

USB Transmit Interrupt Enable (USBTXIE)

Base 0x4005.0000
Offset 0x006
Type R/W, reset 0x000F



Bit/Field	Name	Type	Reset	Description
15:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EP3	R/W	1	TX Endpoint 3 Interrupt Enable
2	EP2	R/W	1	TX Endpoint 2 Interrupt Enable
1	EP1	R/W	1	TX Endpoint 1 Interrupt Enable
0	EP0	R/W	1	TX and RX Endpoint 0 Interrupt Enable

Register 6: USB Receive Interrupt Enable (USBRXIE), offset 0x008

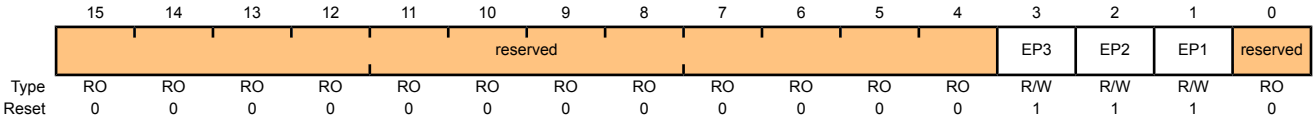
Host

Device

USBRXIE is a 16-bit register that provides interrupt enable bits for the interrupts in **USBRXIS**. When a bit in **USBRXIE** is set to 1, the USB interrupt to the processor is asserted when the corresponding interrupt bit in the **USBRXIS** register is set. When a bit is cleared to 0, the interrupt in **USBRXIS** is still set but the USB interrupt to the processor is not asserted. On reset, the bits corresponding to receive endpoints 1-3 are set to 1, while the remaining bits are set to 0.

USB Receive Interrupt Enable (USBRXIE)

Base 0x4005.0000
Offset 0x008
Type R/W, reset 0x000E



Bit/Field	Name	Type	Reset	Description
15:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EP3	R/W	1	RX Endpoint 3 Interrupt Enable
2	EP2	R/W	1	RX Endpoint 2 Interrupt Enable
1	EP1	R/W	1	RX Endpoint 1 Interrupt Enable
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 7: USB General Interrupt Status (USBIS), offset 0x00A

Host

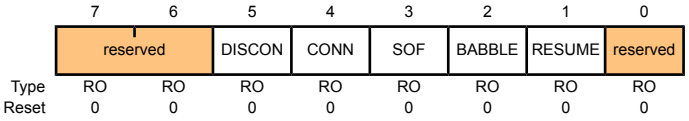
Device

USBIS is an 8-bit read-only register that indicates which USB interrupts are currently active. All active interrupts are cleared when this register is read.

Host Mode

USB General Interrupt Status (USBIS)

Base 0x4005.0000
Offset 0x00A
Type RO, reset 0x00

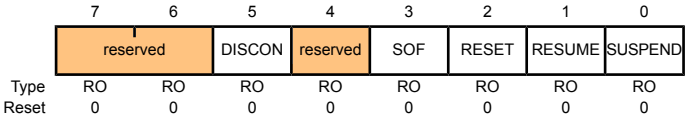


Bit/Field	Name	Type	Reset	Description
7:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	DISCON	RO	0	Session Disconnect Set when a device disconnect is detected.
4	CONN	RO	0	Session Connect Set when a device connection is detected.
3	SOF	RO	0	Start of Frame Set when a new frame starts.
2	BABBLE	RO	0	Babble Detected Set when babble is detected. Only active after first SOF has been sent.
1	RESUME	RO	0	Resume Signal Detected Set when Resume signaling is detected on the bus while the USB controller is in Suspend mode. This can only be used if the USB's system clock is enabled. If the user disables the clock programming, the **USBDRCRIS**, **USBDRCIM**, and **USBISC** registers should be used.
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Device Mode

USB General Interrupt Status (USBIS)

Base 0x4005.0000
Offset 0x00A
Type RO, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	DISCON	RO	0	Session Disconnect Set when a session ends. Valid at all transaction speeds.
4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	SOF	RO	0	Start of Frame Set when a new frame starts.
2	RESET	RO	0	Reset Signal Detected Set when Reset signaling is detected on the bus.
1	RESUME	RO	0	Resume Signal Detected Set when Resume signaling is detected on the bus while the USB controller is in Suspend mode. This can only be used if the USB's system clock is enabled. If the user disables the clock programming, the **USBDRCRIS**, **USBDRCIM**, and **USBISC** registers should be used.
0	SUSPEND	RO	0	Suspend Signal Detected Set when Suspend signaling is detected on the bus.

Register 8: USB Interrupt Enable (USBIE), offset 0x00B

Host

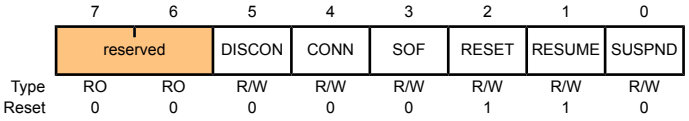
Device

USBIE is an 8-bit register that provides interrupt enable bits for each of the interrupts in **USBIS**. By default, interrupt 1 and 2 are enabled.

Host Mode

USB Interrupt Enable (USBIE)

Base 0x4005.0000
Offset 0x00B
Type R/W, reset 0x06

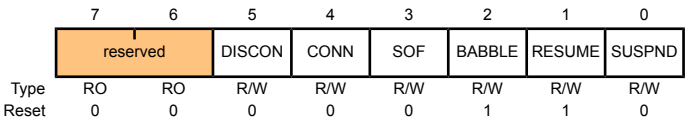


Bit/Field	Name	Type	Reset	Description
7:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	DISCON	R/W	0	Enable Disconnect Interrupt Set by CPU to enable DISCON in **USBIS**.
4	CONN	R/W	0	Enable Connect Interrupt Set by CPU to enable CONN in **USBIS**.
3	SOF	R/W	0	Enable Start-of-Frame Interrupt Set by CPU to enable SOF in **USBIS**.
2	RESET	R/W	1	Enable Reset Interrupt Set by CPU to enable RESET in **USBIS**.
1	RESUME	R/W	1	Enable Resume Interrupt Set by CPU to enable RESUME in **USBIS**.
0	SUSPND	R/W	0	Enable Suspend Interrupt Set by CPU to enable SUSPEND in **USBIS**.

Device Mode

USB Interrupt Enable (USBIE)

Base 0x4005.0000
Offset 0x00B
Type R/W, reset 0x06



Bit/Field	Name	Type	Reset	Description
7:6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	DISCON	R/W	0	Enable Disconnect Interrupt Set by CPU to enable DISCON in **USBIS**.
4	CONN	R/W	0	Enable Connect Interrupt Set by CPU to enable CONN in **USBIS**.
3	SOF	R/W	0	Enable Start-of-Frame Interrupt Set by CPU to enable SOF in **USBIS**.
2	BABBLE	R/W	1	Enable Babble Interrupt Set by CPU to enable BABBLE in **USBIS**.
1	RESUME	R/W	1	Enable Resume Interrupt Set by CPU to enable RESUME in **USBIS**.
0	SUSPND	R/W	0	Enable Suspend Interrupt Set by CPU to enable SUSPEND in **USBIS**.

Register 9: USB Frame Value (USBFRAME), offset 0x00C

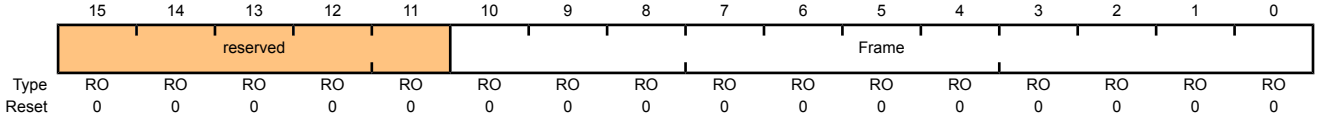
Host

Device

USBFRAME is a 16-bit read-only register that holds the last received frame number.

USB Frame Value (USBFRAME)

Base 0x4005.0000
Offset 0x00C
Type RO, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:11	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
10:0	Frame	RO	0x00	Frame Number

Register 10: USB Endpoint Index (USBEPIDX), offset 0x0E

Host

Device

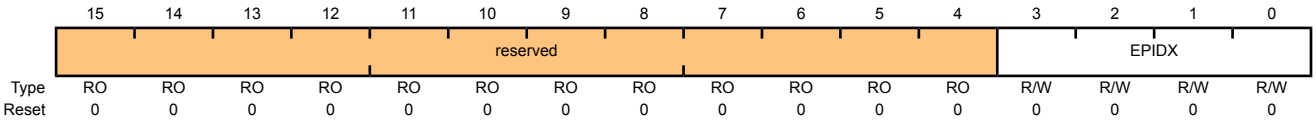
Each endpoint's buffer can be accessed by configuring a FIFO size and starting address. The **USBEPIDX** 16-bit register is used with the **USBTXFIFOSZ**, **USBRXFIFOSZ**, **USBTXFIFOADD**, and **USBRXFIFOADD** registers.

USB Endpoint Index (USBEPIDX)

Base 0x4005.0000

Offset 0x0E

Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	EPIDX	R/W	0x00	Endpoint Index This sets which endpoint is accessed when reading or writing to one of the USB controller's indexed registers.

Register 11: USB Test Mode (USBTEST), offset 0x00F

Host

Device

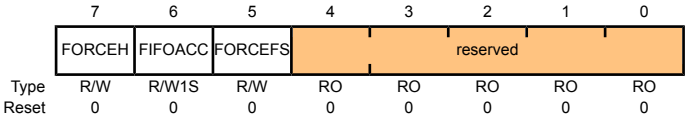
USBTESTMODE is an 8-bit register that is primarily used to put the USB controller into one of the four test modes for operation described in the *USB 2.0 specification*, in response to a SET FEATURE: USBTESTMODE command. It is not used in normal operation.

Note: Only one of these bits should be set at any time.

Host Mode

USB Test Mode (USBTEST)

Base 0x4005.0000
Offset 0x00F
Type R/W, reset 0x00

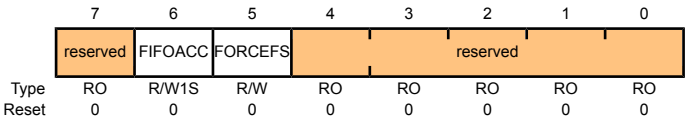


Bit/Field	Name	Type	Reset	Description
7	FORCEH	R/W	0	Force Host Mode The CPU sets this bit to instruct the core to enter Host mode when the Session bit is set, regardless of whether it is connected to any peripheral. The state of the USBD+ and USBD- are ignored. The core then remains in Host mode until the SESSION bit is cleared, even if a device is disconnected, and if the FORCEH bit remains set, re-enters Host mode the next time the SESSION bit is set. While in this mode, status of the bus connection may be read from the DEV bit of the **USBDEVCTL** register. The operating speed is determined from the FORCEFS bit.
6	FIFOACC	R/W1S	0	FIFO Access The CPU sets this bit to transfer the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO. It is cleared automatically.
5	FORCEFS	R/W	0	Force Full-Speed Mode The CPU sets this bit to force the USB controller into Full-Speed mode when it receives a USB reset. When 0, the USB controller operates at Low Speed.
4:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Device Mode

USB Test Mode (USBTEST)

Base 0x4005.0000
Offset 0x00F
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6	FIFOACC	R/W1S	0	FIFO Access The CPU sets this bit to transfer the packet in the endpoint 0 transmit FIFO to the endpoint 0 receive FIFO. It is cleared automatically.
5	FORCEFS	R/W	0	Force Full Speed The CPU sets this bit to force the USB controller into Full-Speed mode when it receives a USB reset. When 0, the USB controller operates at Low Speed.
4:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 12: USB FIFO Endpoint 0 (USBFIFO0), offset 0x020

Register 13: USB FIFO Endpoint 1 (USBFIFO1), offset 0x024

Register 14: USB FIFO Endpoint 2 (USBFIFO2), offset 0x028

Register 15: USB FIFO Endpoint 3 (USBFIFO3), offset 0x02C

Host

Device

These 32-bit registers provide an address for CPU access to the FIFOs for each endpoint. Writing to these addresses loads data into the Transmit FIFO for the corresponding endpoint. Reading from these addresses unloads data from the Receive FIFO for the corresponding endpoint.

Transfers to and from FIFOs may be 8-bit, 16-bit or 32-bit as required, and any combination of access is allowed provided the data accessed is contiguous. All transfers associated with one packet must be of the same width so that the data is consistently byte-, word- or double-word-aligned. However, the last transfer may contain fewer bytes than the previous transfers in order to complete an odd-byte or odd-word transfer.

Depending on the size of the FIFO and the expected maximum packet size, the FIFOs support either single-packet or double-packet buffering. Burst writing of multiple packets is not supported as flags need to be set after each packet is written.

Following a STALL response or a transmit error on endpoint 1–3, the associated FIFO is completely flushed.

USB FIFO Endpoint 0 (USBFIFO0)

Base 0x4005.0000
Offset 0x020
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	EPDATA															
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
	EPDATA															
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	EPDATA	R/W	0x00	Endpoint Data Writing to this register loads the data into the Transmit FIFO and reading unloads data from the Receive FIFO.

Register 16: USB Device Control (USBDEVCTL), offset 0x060

Host

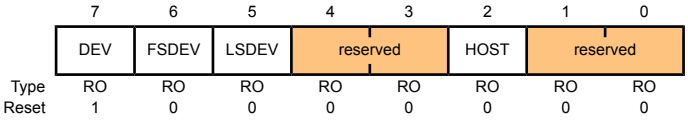
Device

USBDEVCTL provides the status information for the current operating mode (host or device) of the USB controller. If the USB controller is in host mode, this register also indicates if a full- or low-speed device has been connected.

Host Mode

USB Device Control (USBDEVCTL)

Base 0x4005.0000
Offset 0x060
Type R/W, reset 0x80

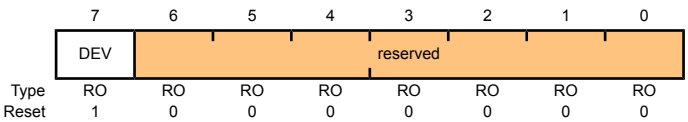


Bit/Field	Name	Type	Reset	Description
7	DEV	RO	1	Device Mode When set, this bit indicates the controller is operating as a device. **Note:** This value is only valid while a session is in progress.
6	FSDEV	RO	0	Full-Speed Device Detected This read-only bit is set when a full-speed device has been detected on the port.
5	LSDEV	RO	0	Low-Speed Device Detected This read-only bit is set when a low-speed device has been detected on the port.
4:3	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	HOST	RO	0	Host Mode This read-only bit is set when the USB controller is acting as a Host.
1:0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Device Mode

USB Device Control (USBDEVCTL)

Base 0x4005.0000
Offset 0x060
Type R/W, reset 0x80



Bit/Field	Name	Type	Reset	Description
7	DEV	RO	1	Device Mode When set, this bit indicates the controller is operating as a device. **Note:** This value is only valid while a session is in progress.
6:0	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 17: USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ), offset 0x062

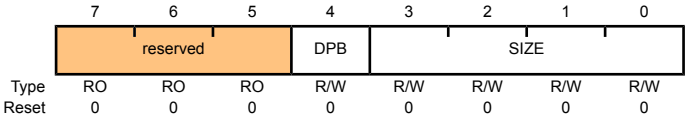
Register 18: USB Receive Dynamic FIFO Sizing (USBRXFIFOSZ), offset 0x063

Host

These 8-bit registers allow the selected TX/RX endpoint FIFOs to be dynamically sized. **USBEPIDX** is used to configure each transmit endpoint's FIFO size.

Device

USB Transmit Dynamic FIFO Sizing (USBTXFIFOSZ)
Base 0x4005.0000
Offset 0x062
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:5	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4	DPB	R/W	0	Double Packet Buffer Support Defines whether double-packet buffering is supported. When 1, double-packet buffering is supported. When 0, only single-packet buffering is supported.
3:0	SIZE	R/W	0x0	Max Packet Size Maximum packet size to be allowed for (*before* any splitting within the FIFO of bulk/high-bandwidth packets prior to transmission. If DPB = 0, the FIFO also is this size; if DPB = 1, the FIFO is twice this size.

Value	Packet Size (Bytes)
0x0	8
0x1	16
0x2	32
0x3	64
0x4	128
0x5	256
0x6	512
0x7	1024
0x8	2048
0x9-0xF	Reserved

Register 19: USB Transmit FIFO Start Address (USBTXFIFOADD), offset 0x064

Register 20: USB Receive FIFO Start Address (USBRXFIFOADD), offset 0x066

Host

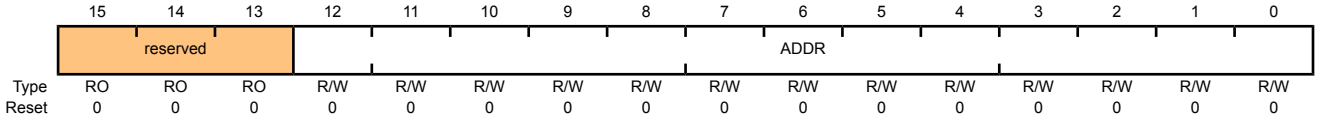
Device

USBTXFIFOADD is a 16-bit register that controls the start address of the selected transmit endpoint FIFO. **USBRXFIFOADD** is a 14-bit register that controls the start address of the selected receive endpoint FIFO.

USB Transmit FIFO Start Address (USBTXFIFOADD)

Base 0x4005.0000

Offset 0x064
Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:13	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12:0	ADDR	R/W	0x00	Transmit/Receive Start Address Start address of the endpoint FIFO in units of 8 bytes.

Value	Start Address
0x0	0
0x1	8
0x2	16
0x3	32
0x4	64
0x5	128
0x6	256
0x7	512
0x8	1024
0x9	2048
0xA-0x1FFF	Reserved

Register 21: USB Connect Timing (USBCONTIM), offset 0x07A

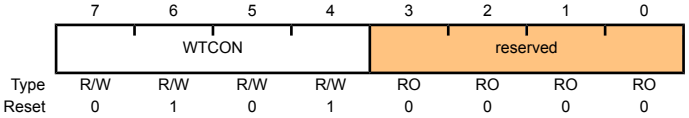
Host

This 8-bit configuration register allows some delays to be specified.

Device

USB Connect Timing (USBCONTIM)

Base 0x4005.0000
Offset 0x07A
Type R/W, reset 0x5C



Bit/Field	Name	Type	Reset	Description
7:4	WTCON	R/W	0x5	Connect Wait Sets the wait to be applied to allow for the user's connect/disconnect filter, in units of 533.3 ns. (The default setting corresponds to 2.667µs.)
3:0	reserved	RO	0	Software should not rely on the value of 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: USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF), offset 0x07D

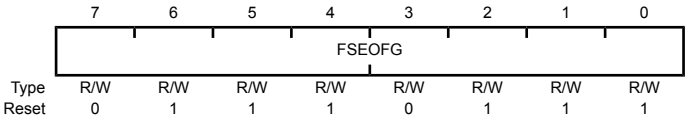
Host

This 8-bit configuration register sets the minimum time gap that is to be allowed between the start of the last transaction and the EOF for full-speed transactions.

Device

USB Full-Speed Last Transaction to End of Frame Timing (USBFSEOF)

Base 0x4005.0000
Offset 0x07D
Type R/W, reset 0x77



Bit/Field	Name	Type	Reset	Description
7:0	FSEOFG	R/W	0x77	Full-Speed End-of-Frame Gap Used during full-speed transactions, to set the gap between the last transaction and the End-of-Frame (EOF), in units of 533.3 ns. The default corresponds to 63.46 µs.

Register 23: USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF), offset 0x07E

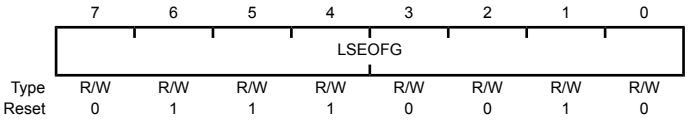
Host

This 8-bit configuration register sets the minimum time gap that is to be allowed between the start of the last transaction and the EOF for low-speed transactions.

Device

USB Low-Speed Last Transaction to End of Frame Timing (USBLSEOF)

Base 0x4005.0000
Offset 0x07E
Type R/W, reset 0x72



Bit/Field	Name	Type	Reset	Description
7:0	LSEOFG	R/W	0x72	Low-Speed End-of-Frame Gap Used during low-speed transactions, to set the gap between the last transaction and the End-of-Frame (EOF), in units of 1.067 µs. The default corresponds to 121.6 µs.

Register 24: USB Transmit Functional Address Endpoint 0 (USBTXFUNCADDR0), offset 0x080

Register 25: USB Transmit Functional Address Endpoint 1 (USBTXFUNCADDR1), offset 0x088

Register 26: USB Transmit Functional Address Endpoint 2 (USBTXFUNCADDR2), offset 0x090

Register 27: USB Transmit Functional Address Endpoint 3 (USBTXFUNCADDR3), offset 0x098

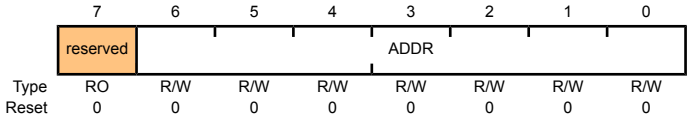
Host

USBTXFUNCADDRn is an 8-bit read/write register that records the address of the target function that is to be accessed through the associated endpoint (EP*n*). **USBTXFUNCADDRn** needs to be defined for each transmit endpoint that is used.

Note: **USBTXFUNCADDR0** is used for both receive and transmit for endpoint 0.

USB Transmit Functional Address Endpoint 0 (USBTXFUNCADDR0)

Base 0x4005.0000
Offset 0x080
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	ADDR	R/W	0x00	Device Address USB bus address for the target device.

Register 28: USB Transmit Hub Address Endpoint 0 (USBTXHUBADDR0), offset 0x082

Register 29: USB Transmit Hub Address Endpoint 1 (USBTXHUBADDR1), offset 0x08A

Register 30: USB Transmit Hub Address Endpoint 2 (USBTXHUBADDR2), offset 0x092

Register 31: USB Transmit Hub Address Endpoint 3 (USBTXHUBADDR3), offset 0x09A

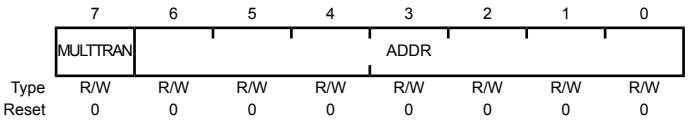
Host

USBTXHUBADDRn is an 8-bit read/write register that, like **USBTXHUBPORTn**, only needs to be written when a USB device is connected to transmit endpoint EPn via a USB 2.0 hub. This register records the address of that USB 2.0 hub through which the target associated with the endpoint is accessed. This information, together with the hub port in **USBTXHUBPORTn**, allows the USB controller to support split transactions.

Note: **USBTXHUBADDR0** is used for both receive and transmit for endpoint 0.

USB Transmit Hub Address Endpoint 0 (USBTXHUBADDR0)

Base 0x4005.0000
Offset 0x082
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	MULTTRAN	R/W	0	Multiple Translators Indicates whether the hub has multiple transaction translators. Clear to 0 if single transaction translator; set to 1 if multiple transaction translators.
6:0	ADDR	R/W	0x00	Hub Address USB bus address for the USB 2.0 hub.

Register 32: USB Transmit Hub Port Endpoint 0 (USBTXHUBPORT0), offset 0x083

Register 33: USB Transmit Hub Port Endpoint 1 (USBTXHUBPORT1), offset 0x08B

Register 34: USB Transmit Hub Port Endpoint 2 (USBTXHUBPORT2), offset 0x093

Register 35: USB Transmit Hub Port Endpoint 3 (USBTXHUBPORT3), offset 0x09B

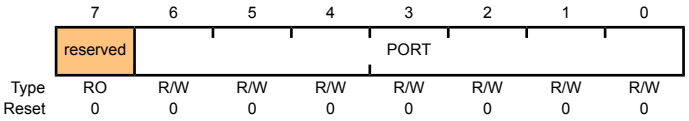
Host

USBTXHUBPORTn is an 8-bit read/write register that, like **USBTXHUBADDRn**, only needs to be written when a full- or low-speed device is connected to transmit endpoint EPn via a USB 2.0 hub. This register records the port of that USB 2.0 hub through which the target associated with the endpoint is accessed. This information, together with the hub address in **USBTXHUBADDRn**, allows the USB controller to support split transactions.

Note: **USBTXHUBPORT0** is used for both receive and transmit for endpoint 0.

USB Transmit Hub Port Endpoint 0 (USBTXHUBPORT0)

Base 0x4005.0000
Offset 0x083
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	PORT	R/W	0x00	Hub Port USB hub port number.

Register 36: USB Receive Functional Address Endpoint 1 (USBRXFUNCADDR1), offset 0x08C

Register 37: USB Receive Functional Address Endpoint 2 (USBRXFUNCADDR2), offset 0x094

Register 38: USB Receive Functional Address Endpoint 3 (USBRXFUNCADDR3), offset 0x09C

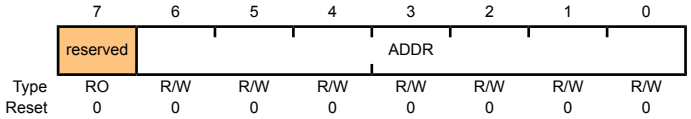
Host

USBRXFUNCADDRn is an 8-bit read/write register that records the address of the target function that is to be accessed through the associated endpoint (EP*n*). **USBRXFUNCADDRn** needs to be defined for each receive endpoint that is used.

Note: **USBTXFUNCADDR0** is used for both receive and transmit for endpoint 0.

USB Receive Functional Address Endpoint 1 (USBRXFUNCADDR1)

Base 0x4005.0000
Offset 0x08C
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	ADDR	R/W	0x00	Device Address USB bus address for the target device.

Register 39: USB Receive Hub Address Endpoint 1 (USBRXHUBADDR1), offset 0x08E

Register 40: USB Receive Hub Address Endpoint 2 (USBRXHUBADDR2), offset 0x096

Register 41: USB Receive Hub Address Endpoint 3 (USBRXHUBADDR3), offset 0x09E

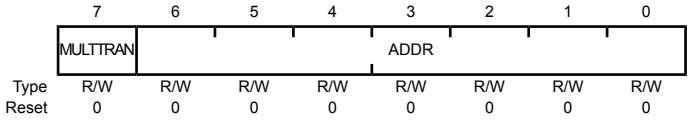
Host

USBRXHUBADDRn is an 8-bit read/write register that, like **USBRXHUBPORTn**, only needs to be written when a full- or low-speed device is connected to receive endpoint EPn via a USB 2.0 hub. This register records the address of that USB 2.0 hub through which the target associated with the endpoint is accessed. This information, together with the hub port in **USBRXHUBPORTn**, allows the USB controller to support split transactions.

Note: **USBTXHUBADDR0** is used for both receive and transmit for endpoint 0.

USB Receive Hub Address Endpoint 1 (USBRXHUBADDR1)

Base 0x4005.0000
Offset 0x08E
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	MULTTRAN	R/W	0	Multiple Translators Indicates whether the hub has multiple transaction translators. Clear to 0 if single transaction translator; set to 1 if multiple transaction translators.
6:0	ADDR	R/W	0x00	Hub Address USB bus address for the USB 2.0 hub.

Register 42: USB Receive Hub Port Endpoint 1 (USBRXHUBPORT1), offset 0x08F

Register 43: USB Receive Hub Port Endpoint 2 (USBRXHUBPORT2), offset 0x097

Register 44: USB Receive Hub Port Endpoint 3 (USBRXHUBPORT3), offset 0x09F

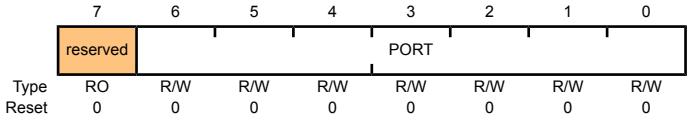
Host

USBRXHUBPORTn is an 8-bit read/write register that, like **USBRXHUBADDRn**, only needs to be written when a full- or low-speed device is connected to receive endpoint EPn via a USB 2.0 hub. This register records the port of that USB 2.0 hub through which the target associated with the endpoint is accessed. This information, together with the hub address in **USBTXHUBADDRn**, allows the USB controller to support split transactions.

Note: **USBTXHUBPORT0** is used for both receive and transmit for endpoint 0.

USB Receive Hub Port Endpoint 1 (USBRXHUBPORT1)

Base 0x4005.0000
Offset 0x08F
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	PORT	R/W	0x00	Hub Port USB hub port number.

Register 45: USB Maximum Transmit Data Endpoint 1 (USBTXMAXP1), offset 0x110

Register 46: USB Maximum Transmit Data Endpoint 2 (USBTXMAXP2), offset 0x120

Register 47: USB Maximum Transmit Data Endpoint 3 (USBTXMAXP3), offset 0x130

Host

The **USBTXMAXPn** 16-bit register defines the maximum amount of data that can be transferred through the transmit endpoint in a single operation.

Device

Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the *USB Specification* on packet sizes for bulk, interrupt and isochronous transfers in full-speed operation.

The MULT bit field contains the multiplication factor for the number of bytes in a given transaction. For a single 64-byte bulk transfer, the multiplication factor is 1 so MULT should be written with 0. If packet splitting is used, the multiplication factor allows for more than one transfer to be loaded into the FIFO. A multiplication factor of 2 (MULT written to 1) allows two 64-byte packets to be written in this endpoint's FIFO.

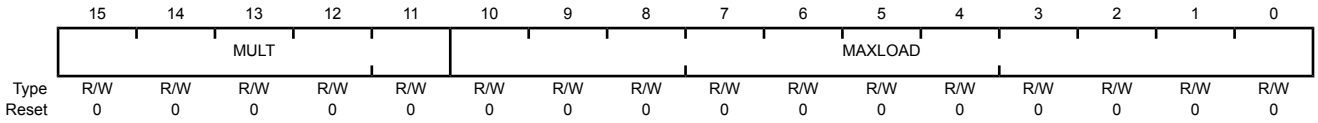
The total amount of data represented by the value written to this register (specified payload × *m*) must not exceed the FIFO size for the transmit endpoint, and should not exceed half the FIFO size if double-buffering is required.

If this register is changed after packets have been sent from the endpoint, the transmit endpoint FIFO should be completely flushed (using the FLUSH bit in **USBTXCSRL1n**) after writing the new value to this register.

Note: **USBTXMAXPn** must be set to an even number of bytes for proper interrupt generation in DMA Mode 1.

USB Maximum Transmit Data Endpoint 1 (USBTXMAXP1)

Base 0x4005.0000
Offset 0x110
Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:11	MULT	R/W	0x00	Multiplier Defines the maximum number of USB packets (that is, packets for transmission over the USB) of the specified payload into which a single data packet placed in the FIFO should be split, prior to transfer. The value written to this register is one less than the desired multiplier. For example, a value of 0 is a multiplier of 1.
10:0	MAXLOAD	R/W	0x00	Maximum Payload The maximum payload in bytes per transaction.

Register 48: USB Control and Status Endpoint 0 Low (USBCSRL0), offset 0x102

Host

Device

USBCSRL0 is an 8-bit register that provides control and status bits for endpoint 0.

Host Mode

USB Control and Status Endpoint 0 Low (USBCSRL0)

Base 0x4005.0000
Offset 0x102
Type W1C, reset 0x00

	7	6	5	4	3	2	1	0
	NAKTO	STATUS	REQPKT	ERROR	SETUP	STALLED	TXRDY	RXRDY
Type	R/W0C	R/W	R/W	R/W0C	R/W1S	R/W0C	R/W1S	R/W0C
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	NAKTO	R/W0C	0	NAK Timeout This bit is set by the USB controller when endpoint 0 is halted following the receipt of NAK responses for longer than the time set by the **USBNAKLMT** register. The CPU should clear this bit by writing a 0 to it to allow the endpoint to continue.
6	STATUS	R/W	0	Status Packet The CPU sets this bit at the same time as the TXRDY or REQPKT bit is set, to perform a status stage transaction. Setting this bit ensures DT is set to 1 so that a DATA1 packet is used for the Status Stage transaction.
5	REQPKT	R/W	0	Request Packet The CPU sets this bit to request an IN transaction. It is cleared when RXRDY is set.
4	ERROR	R/W0C	0	Error This bit is set by the USB controller when three attempts have been made to perform a transaction with no response from the peripheral. The CPU should clear this bit. An interrupt is generated when this bit is set.
3	SETUP	R/W1S	0	Setup Packet The CPU sets this bit, at the same time as the TXRDY bit is set, to send a SETUP token instead of an OUT token for the transaction. This always resets the data toggle and sends a DATA0 packet.
2	STALLED	R/W0C	0	Endpoint Stalled This bit is set when a STALL handshake is received. The CPU should clear this bit.

Bit/Field	Name	Type	Reset	Description
1	TXRDY	R/W1S	0	Transmit Packet Ready The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is also generated at this point.
0	RXRDY	R/W0C	0	Receive Packet Ready This bit is set when a data packet has been received. An interrupt is generated when this bit is set. The CPU should clear this bit, by writing a 0 when the packet has been read from the FIFO. This acknowledges that data has been read from the FIFO.

Device Mode

USB Control and Status Endpoint 0 Low (USBCSRL0)

Base 0x4005.0000
Offset 0x102
Type W1C, reset 0x00

	7	6	5	4	3	2	1	0
	SETENDC	RXRDYC	STALL	SETEND	DATAEND	STALLED	TXRDY	RXRDY
Type	W1C	W1C	W1C	RO	W1C	R/W0C	R/W1S	RO
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	SETENDC	W1C	0	Setup End Clear The CPU writes a 1 to this bit to clear the SETEND bit.
6	RXRDYC	W1C	0	RXRDY Clear The CPU writes a 1 to this bit to clear the RXRDY bit.
5	STALL	W1C	0	Send Stall The CPU writes a 1 to this bit to terminate the current transaction. The STALL handshake is transmitted, and then this bit is cleared automatically.
4	SETEND	RO	0	Setup End This bit is set when a control transaction ends before the DataEnd bit has been set. An interrupt is generated and the FIFO flushed at this time. The bit is cleared by the CPU writing a 1 to the SETENDC bit.
3	DATAEND	W1C	0	Data End The CPU sets this bit: ■ When setting TXRDY for the last data packet ■ When clearing RXRDY after unloading the last data packet ■ When setting TXRDY for a zero-length data packet It is cleared automatically.

Bit/Field	Name	Type	Reset	Description
2	STALLED	R/W0C	0	Endpoint Stalled This bit is set when a STALL handshake is transmitted. The CPU should clear this bit by writing a 0. This bit can only be cleared. Setting this bit does nothing.
1	TXRDY	R/W1S	0	Transmit Packet Ready The CPU writes a 1 to this bit after loading a data packet into the FIFO. It is cleared automatically when the data packet has been transmitted. An interrupt is also generated at this point.
0	RXRDY	RO	0	Receive Packet Ready This bit is set when a data packet has been received. An interrupt is generated when this bit is set. The CPU clears this bit by setting the RXRDYC bit.

Register 49: USB Control and Status Endpoint 0 High (USBCSRH0), offset 0x103

Host

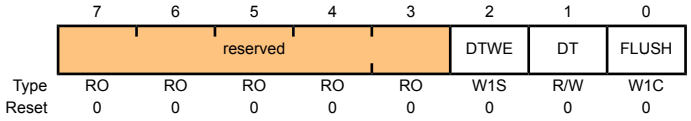
Device

USBSR0H is an 8-bit register that provides control and status bits for endpoint 0.

Host Mode

USB Control and Status Endpoint 0 High (USBCSRH0)

Base 0x4005.0000
Offset 0x103
Type W1C, reset 0x00



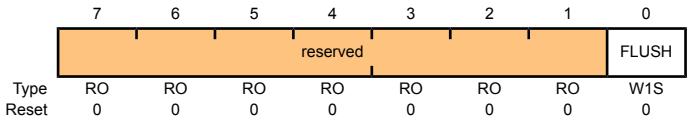
Bit/Field	Name	Type	Reset	Description
7:3	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
2	DTWE	W1S	0	Data Toggle Write Enable The CPU writes a 1 to this bit to enable the current state of the endpoint 0 data toggle to be written (see DT bit). This bit is automatically cleared once the new value is written.
1	DT	R/W	0	Data Toggle When read, this bit indicates the current state of the endpoint 0 data toggle. If DTWE is High, this bit may be written with the required setting of the data toggle. If DTWE is Low, this cannot be written.
0	FLUSH	W1C	0	Flush FIFO The CPU writes a 1 to this bit to flush the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXRDY/RXRDY bit is cleared.

Important: FLUSH should only be used when TXRDY/RXRDY is set. At other times, it may cause data to be corrupted.

Device Mode

USB Control and Status Endpoint 0 High (USBCSRH0)

Base 0x4005.0000
Offset 0x103
Type W1C, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	FLUSH	W1S	0	Flush FIFO The CPU writes a 1 to this bit to flush the next packet to be transmitted/read from the endpoint 0 FIFO. The FIFO pointer is reset and the TXRDY/RXRDY bit is cleared. **Important:** FLUSH should only be used when TXRDY/RXRDY is set. At other times, it may cause data to be corrupted.

Register 50: USB Receive Byte Count Endpoint 0 (USBCOUNT0), offset 0x108

Host

Device

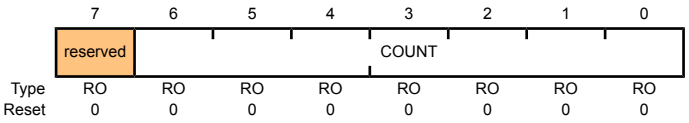
USBCOUNT0 is an 8-bit read-only register that indicates the number of received data bytes in the endpoint 0 FIFO. The value returned changes as the contents of the FIFO change and is only valid while RXRDY is set.

USB Receive Byte Count Endpoint 0 (USBCOUNT0)

Base 0x4005.0000

Offset 0x108

Type RO, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
6:0	COUNT	RO	0x00	Count Count is a read-only value that indicates the number of received data bytes in the endpoint 0 FIFO.

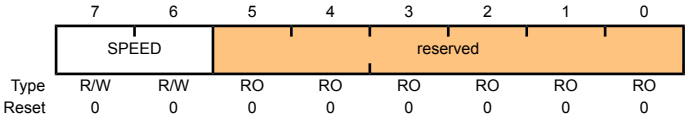
Register 51: USB Type Endpoint 0 (USBTYPE0), offset 0x10A

Host

This is an 8-bit register that should be written with the operating speed of the targeted device being communicated with using endpoint 0.

USB Type Endpoint 0 (USBTYPE0)

Base 0x4005.0000
Offset 0x10A
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:6	SPEED	R/W	0x00	Operating Speed Operating speed of the target device. If selected, the target is assumed to have the same connection speed as the core. Value Description 00 Reserved 01 Reserved 10 Full 11 Low
5:0	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 52: USB NAK Limit (USBNAKLMT), offset 0x10B

Host

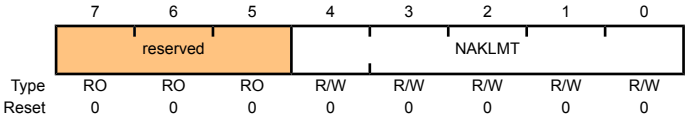
USBNAKLMT is an 8-bit register that sets the number of frames after which endpoint 0 should time out on receiving a stream of NAK responses. (Equivalent settings for other endpoints can be made through their **USBTXINTERVALn** and **USBRXINTERVALn** registers.)

The number of frames selected is $2^{(m-1)}$ (where *m* is the value set in the register, with valid values of 2–16). If the host receives NAK responses from the target for more frames than the number represented by the limit set in this register, the endpoint is halted.

Note: A value of 0 or 1 disables the NAK timeout function.

USB NAK Limit (USBNAKLMT)

Base 0x4005.0000
Offset 0x10B
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:5	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
4:0	NAKLMT	R/W	0x00	EP0 NAK Limit Number of frames after receiving a stream of NAK responses.

Register 53: USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1), offset 0x112

Register 54: USB Transmit Control and Status Endpoint 2 Low (USBTXCSRL2), offset 0x122

Register 55: USB Transmit Control and Status Endpoint 3 Low (USBTXCSRL3), offset 0x132

Host

Device

USBTXCSRLn is an 8-bit register that provides control and status bits for transfers through the currently selected transmit endpoint.

Host Mode

USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1)

Base 0x4005.0000
Offset 0x112
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	NAKTO / INCTX	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
Type	R/W0C	W1S	R/W0C	R/W	W1C	R/W0C	R/W0C	R/W0C
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	NAKTO / INCTX	R/W0C	0	NAK Timeout / Incomplete TX *Bulk endpoints only:* This bit is set when the transmit endpoint is halted following the receipt of NAK responses for longer than the time set as the NAK Limit by the **USBTXINTERVALn** register. The CPU should clear this bit to allow the endpoint to continue. *High-bandwidth interrupt endpoints only:* This bit is set if no response is received from the device to which the packet is being sent.
6	CLRDT	W1S	0	Clear Data Toggle The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.
5	STALLED	R/W0C	0	Endpoint Stalled This bit is set when a STALL handshake is received. When this bit is set, any DMA request that is in progress is stopped, the FIFO is completely flushed, and the TXRDY bit is cleared. The CPU should clear this bit.
4	SETUP	R/W	0	Setup Packet The CPU sets this bit, at the same time as the TXRDY bit is set, to send a SETUP token instead of an OUT token for the transaction. **Note:** Setting this bit also clears DT.

Bit/Field	Name	Type	Reset	Description
3	FLUSH	W1C	0	Flush FIFO The CPU writes a 1 to this bit to flush the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset, the TXRDY bit is cleared, and an interrupt is generated. FLUSH may be set simultaneously with TXRDY to abort the packet that is currently being loaded into the FIFO. **Note:** FLUSH should only be used when TXRDY is set. At other times, it may cause data to be corrupted. Also note that, if the FIFO is double-buffered, FLUSH may need to be set twice to completely clear the FIFO.
2	ERROR	R/W0C	0	Error The USB sets this bit when three attempts have been made to send a packet and no handshake packet has been received. When the bit is set, an interrupt is generated, TXRDY is cleared, and the FIFO is completely flushed. The CPU should clear this bit. **Note:** This is valid only when the endpoint is operating in Bulk or Interrupt mode.
1	FIFONE	R/W0C	0	FIFO Not Empty The USB controller sets this bit when there is at least one packet in the transmit FIFO.
0	TXRDY	R/W0C	0	Transmit Packet Ready The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is generated at this point. TXRDY is also automatically cleared prior to loading a second packet into a double-buffered FIFO.

Device Mode

USB Transmit Control and Status Endpoint 1 Low (USBTXCSRL1)

Base 0x4005.0000
Offset 0x112
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	INCTX	CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
Type	R/W0C	W1S	R/W0C	R/W	W1C	R/W0C	R/W0C	R/W1S
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	INCTX	R/W0C	0	Incomplete Transmit When the endpoint is being used for high-bandwidth isochronous transfers, this bit is set to indicate where a large packet has been split into 2 or 3 packets for transmission but insufficient IN tokens have been received to send all the parts. **Note:** Only valid for isochronous transfers.
6	CLRDT	W1S	0	Clear Data Toggle The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.

Bit/Field	Name	Type	Reset	Description
5	STALLED	R/W0C	0	Endpoint Stalled This bit is set when a STALL handshake is transmitted. The FIFO is flushed and the TXRDY bit is cleared. The CPU should clear this bit.
4	STALL	R/W	0	Send Stall The CPU writes a 1 to this bit to issue a STALL handshake to an IN token. The CPU clears this bit to terminate the stall condition. **Note:** This bit has no effect in isochronous transfers.
3	FLUSH	W1C	0	Flush FIFO The CPU writes a 1 to this bit to flush the latest packet from the endpoint transmit FIFO. The FIFO pointer is reset, the TXRDY bit is cleared, and an interrupt is generated. This bit may be set simultaneously with TXRDY to abort the packet that is currently being loaded into the FIFO. **Note:** FLUSH should only be used when TXRDY is set. At other times, it may cause data to be corrupted. Also note that, if the FIFO is double-buffered, FLUSH may need to be set twice to completely clear the FIFO.
2	UNDRN	R/W0C	0	Underrun The USB controller sets this bit if an IN token is received when TXRDY is not set. The CPU should clear this bit.
1	FIFONE	R/W0C	0	FIFO Not Empty The USB controller sets this bit when there is at least 1 packet in the transmit FIFO.
0	TXRDY	R/W1S	0	Transmit Packet Ready The CPU sets this bit after loading a data packet into the FIFO. It is cleared automatically when a data packet has been transmitted. An interrupt is generated at this point. TXRDY is also automatically cleared prior to loading a second packet into a double-buffered FIFO.

Register 56: USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1), offset 0x113

Register 57: USB Transmit Control and Status Endpoint 2 High (USBTXCSRH2), offset 0x123

Register 58: USB Transmit Control and Status Endpoint 3 High (USBTXCSRH3), offset 0x133

Host

Device

USBTXCSRHn is an 8-bit register that provides additional control for transfers through the currently selected transmit endpoint.

Host Mode

USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1)

Base 0x4005.0000
Offset 0x113
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	AUTOSET	reserved	MODE	DMAEN	FDT	DMAMOD	DTWE	DT
Type	R/W	RO	R/W	R/W	R/W	R/W	W1S	R/W
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	AUTOSET	R/W	0	Auto Set If the CPU sets this bit, TXRDY is automatically set when data of the maximum packet size (value in **USBTXMAXPn**) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then TXRDY must be set manually. **Note:** This bit should not be set for either high-bandwidth isochronous or high-bandwidth interrupt endpoints.
6	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	MODE	R/W	0	Mode The CPU sets this bit to enable the endpoint direction as TX, and clears it to enable the endpoint direction as RX. **Note:** This bit only has an effect when the same endpoint FIFO is used for both transmit and receive transactions.
4	DMAEN	R/W	0	DMA Request Enable The CPU sets this bit to enable the DMA request for the transmit endpoint.

Bit/Field	Name	Type	Reset	Description
3	FDT	R/W	0	Force Data Toggle The CPU sets this bit to force the endpoint data toggle to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints.
2	DMAMOD	R/W	0	DMA Request Mode The CPU sets this bit to select DMA Request Mode 1 and clears it to select DMA Request Mode 0. **Note:** This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.
1	DTWE	W1S	0	Data Toggle Write Enable The CPU writes a 1 to this bit to enable the current state of the transmit endpoint data toggle to be written (see DT). This bit is automatically cleared once the new value is written.
0	DT	R/W	0	Data Toggle When read, this bit indicates the current state of the transmit endpoint data toggle. If DTWE is High, this bit may be written with the required setting of the data toggle. If DTWE is Low, any value written to this bit is ignored.

Device Mode

USB Transmit Control and Status Endpoint 1 High (USBTXCSRH1)

Base 0x4005.0000
Offset 0x113
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD	reserved	
Type	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	AUTOSET	R/W	0	Auto Set If the CPU sets this bit, TXRDY is automatically set when data of the maximum packet size (value in **USBTXMAXPn**) is loaded into the transmit FIFO. If a packet of less than the maximum packet size is loaded, then TXRDY must be set manually. **Note:** This bit should not be set for either high-bandwidth isochronous or high-bandwidth interrupt endpoints.
6	ISO	R/W	0	ISO The CPU sets this bit to enable the transmit endpoint for isochronous transfers, and clears it to enable the transmit endpoint for bulk or interrupt transfers.

Bit/Field	Name	Type	Reset	Description
5	MODE	R/W	0	Mode The CPU sets this bit to enable the endpoint direction as TX, and clears the bit to enable it as RX. **Note:** This bit only has an effect where the same endpoint FIFO is used for both transmit and receive transactions.
4	DMAEN	R/W	0	DMA Request Enable The CPU sets this bit to enable the DMA request for the transmit endpoint.
3	FDT	R/W	0	Force Data Toggle The CPU sets this bit to force the endpoint data toggle to switch and the data packet to be cleared from the FIFO, regardless of whether an ACK was received. This can be used by interrupt transmit endpoints that are used to communicate rate feedback for isochronous endpoints.
2	DMAMOD	R/W	0	DMA Request Mode The CPU sets this bit to select DMA Request Mode 1 and clears it to select DMA Request Mode 0. **Note:** This bit must not be cleared either before or in the same cycle as the above DMAEN bit is cleared.
1:0	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 59: USB Maximum Receive Data Endpoint 1 (USBRXMAXP1), offset 0x114

Register 60: USB Maximum Receive Data Endpoint 2 (USBRXMAXP2), offset 0x124

Register 61: USB Maximum Receive Data Endpoint 3 (USBRXMAXP3), offset 0x134

Host

The **USBRXMAXPn** 16-bit register defines the maximum amount of data that can be transferred through the selected receive endpoint in a single operation.

Device

Bits 10:0 define (in bytes) the maximum payload transmitted in a single transaction. The value set can be up to 1024 bytes but is subject to the constraints placed by the *USB Specification* on packet sizes for bulk, interrupt and isochronous transfers in full-speed operations.

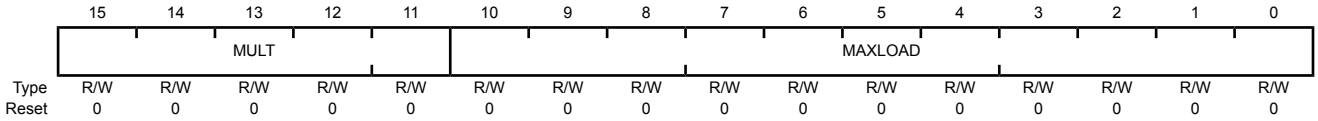
The MULT bit field is for the multiplication factor for the number of bytes in a given transaction. For a single 64-byte bulk transfer, the multiplication factor is 1 so MULT should be written with 0. If packet splitting is used, the multiplication factor allows for more than one transfer to be loaded into the FIFO. A multiplication factor of 2 (MULT written to 1) allows two 64-byte packets to be written in this endpoint's FIFO.

The total amount of data represented by the value written to this register (specified payload × *m*) must not exceed the FIFO size for the receive endpoint, and should not exceed half the FIFO size if double-buffering is required.

Note: **USBRXMAXPn** must be set to an even number of bytes for proper interrupt generation in DMA Mode 1.

USB Maximum Receive Data Endpoint 1 (USBRXMAXP1)

Base 0x4005.0000
Offset 0x114
Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:11	MULT	R/W	0x00	Multiplier Defines the maximum number of USB packets (that is, packets for transmission over the USB) of the specified payload into which a single data packet placed in the FIFO should be split, prior to transfer. The value written to this register is one less than the desired multiplier. For example, a value of 0 is a multiplier of 1.
10:0	MAXLOAD	R/W	0x00	Maximum Payload The maximum payload in bytes per transaction.

Register 62: USB Receive Control and Status Endpoint 1 Low (USBRXCSRL1), offset 0x116

Register 63: USB Receive Control and Status Endpoint 2 Low (USBRXCSRL2), offset 0x126

Register 64: USB Receive Control and Status Endpoint 3 Low (USBRXCSRL3), offset 0x136

Host

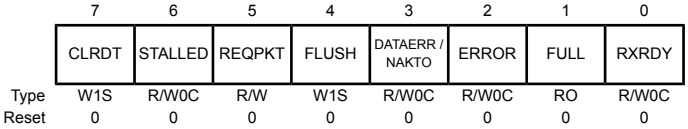
Device

USBRXCSRLn is an 8-bit register that provides control and status bits for transfers through the currently selected receive endpoint.

Host Mode

USB Receive Control and Status Endpoint 1 Low (USBRXCSRL1)

Base 0x4005.0000
Offset 0x116
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7	CLRDT	W1S	0	Clear Data Toggle The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.
6	STALLED	R/W0C	0	Endpoint Stalled When a STALL handshake is received, this bit is set and an interrupt is generated. The CPU should clear this bit.
5	REQPKT	R/W	0	Request Packet The CPU writes a 1 to this bit to request an IN transaction. It is cleared when RXRDY is set.
4	FLUSH	W1S	0	Flush FIFO The CPU writes a 1 to this bit to flush the next packet to be read from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared. **Note:** FLUSH should only be used when RXRDY is set. At other times, it may cause data to be corrupted. Also note that, if the FIFO is double-buffered, FLUSH may need to be set twice to completely clear the FIFO.

Bit/Field	Name	Type	Reset	Description
3	DATAERR / NAKTO	R/W0C	0	Data Error / NAK Timeout When operating in ISO mode, this bit is set when RXRDY is set if the data packet has a CRC or bit-stuff error and cleared when RXRDY is cleared. In Bulk mode, this bit is set when the receive endpoint is halted following the receipt of NAK responses for longer than the time set as the NAK Limit by the **USBRXINTERVALn** register. The CPU should clear this bit to allow the endpoint to continue.
2	ERROR	R/W0C	0	Error The USB sets this bit when three attempts have been made to receive a packet and no data packet has been received. The CPU should clear this bit. An interrupt is generated when the bit is set. **Note:** This bit is only valid when the receive endpoint is operating in Bulk or Interrupt mode. In ISO mode, it always returns zero.
1	FULL	RO	0	FIFO Full This bit is set when no more packets can be loaded into the receive FIFO.
0	RXRDY	R/W0C	0	Receive Packet Ready This bit is set when a data packet has been received. The CPU should clear this bit when the packet has been unloaded from the receive FIFO. An interrupt is generated when the bit is set.

Device Mode

USB Receive Control and Status Endpoint 1 Low (USBRXCSRL1)

Base 0x4005.0000
Offset 0x116
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
Type	W1S	R/W0C	R/W	W1S	RO	R/W0C	RO	R/W0C
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	CLRDT	W1S	0	Clear Data Toggle The CPU writes a 1 to this bit to reset the endpoint data toggle to 0.
6	STALLED	R/W0C	0	Endpoint Stalled This bit is set when a STALL handshake is transmitted. The CPU should clear this bit.
5	STALL	R/W	0	Send Stall The CPU writes a 1 to this bit to issue a STALL handshake. The CPU clears this bit to terminate the stall condition. **Note:** This bit has no effect where the endpoint is being used for isochronous transfers.

Bit/Field	Name	Type	Reset	Description
4	FLUSH	W1S	0	Flush FIFO The CPU writes a 1 to this bit to flush the next packet to be read from the endpoint receive FIFO. The FIFO pointer is reset and the RXRDY bit is cleared. **Note:** The FLUSH bit should only be used when RXRDY is set. At other times, it may cause data to be corrupted. Also note that, if the FIFO is double-buffered, FLUSH may need to be set twice to completely clear the FIFO.
3	DATAERR	RO	0	Data Error This bit is set when RXRDY is set if the data packet has a CRC or bit-stuff error. It is cleared when RXRDY is cleared. **Note:** This bit is only valid when the endpoint is operating in ISO mode. In Bulk mode, it always returns zero.
2	OVER	R/W0C	0	Overrun This bit is set if an OUT packet cannot be loaded into the receive FIFO. The CPU should clear this bit. **Note:** This bit is only valid when the endpoint is operating in ISO mode. In Bulk mode, it always returns zero.
1	FULL	RO	0	FIFO Full This bit is set when no more packets can be loaded into the receive FIFO.
0	RXRDY	R/W0C	0	Receive Packet Ready This bit is set when a data packet has been received. The CPU should clear this bit when the packet has been unloaded from the receive FIFO. An interrupt is generated when the bit is set.

Register 65: USB Receive Control and Status Endpoint 1 High (USBRXCSRH1), offset 0x117

Register 66: USB Receive Control and Status Endpoint 2 High (USBRXCSRH2), offset 0x127

Register 67: USB Receive Control and Status Endpoint 3 High (USBRXCSRH3), offset 0x137

Host

Device

USBRXCSRHn is an 8-bit register that provides additional control and status bits for transfers through the currently selected receive endpoint.

Host Mode

USB Receive Control and Status Endpoint 1 High (USBRXCSRH1)

Base 0x4005.0000
Offset 0x117
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	INCRX
Type	R/W	R/W	R/W	RO	R/W	RO	RO	R/W0C
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	AUTOCL	R/W	0	Auto Clear If the CPU sets this bit, then the RXRDY bit is automatically cleared when a packet of **USBRXMAXPn** bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, RXRDY must be cleared manually. Care must be taken when using a DMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of the RxMaxP bit. **Note:** This bit should not be set for high-bandwidth isochronous endpoints.
6	AUTORQ	R/W	0	Auto Request If the CPU sets this bit, the ReqPkt bit is automatically set when the RXRDY bit is cleared. **Note:** This bit is automatically cleared when a short packet is received.
5	DMAEN	R/W	0	DMA Request Enable The CPU sets this bit to enable the DMA request for the receive endpoint.
4	PIDERR	RO	0	PID Error For ISO transactions, the core sets this bit to indicate a PID error in the received packet. This bit is ignored in bulk or interrupt transactions.

Bit/Field	Name	Type	Reset	Description
3	DMAMOD	R/W	0	DMA Request Mode The CPU sets this bit to select DMA Request Mode 1 and clears it to select DMA Request Mode 0.
2	DTWE	RO	0	Data Toggle Write Enable The CPU writes a 1 to this bit to enable the current state of the endpoint 0 data toggle to be written (see DT). This bit is automatically cleared once the new value is written.
1	DT	RO	0	Data Toggle When read, this bit indicates the current state of the endpoint 0 data toggle. If DTWE is High, this bit may be written with the required setting of the data toggle. If DTWE is Low, any value written to this bit is ignored.
0	INCRX	R/W0C	0	Incomplete Receive This bit is set in a high-bandwidth isochronous or interrupt transfer if the packet received is incomplete. It is cleared when RXRDY is cleared. **Note:** If USB protocols are followed correctly, this bit should never be set. The bit becoming set indicates a failure of the associated peripheral device to behave correctly. (In anything other than isochronous transfer, this bit always returns 0.)

Device Mode

USB Receive Control and Status Endpoint 1 High (USBRXCSRH1)

Base 0x4005.0000
Offset 0x117
Type R/W, reset 0x00

	7	6	5	4	3	2	1	0
	AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD	reserved		INCRX
Type	R/W	R/W	R/W	R/W	R/W	RO	RO	R/W0C
Reset	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
7	AUTOCL	R/W	0	Auto Clear If the CPU sets this bit, then the RXRDY bit is automatically cleared when a packet of RXMaxP bytes has been unloaded from the receive FIFO. When packets of less than the maximum packet size are unloaded, RXRDY must be cleared manually. Care must be taken when using a DMA to unload the receive FIFO as data is read from the receive FIFO in 4 byte chunks regardless of the RxMaxP bit. **Note:** This bit should not be set for high-bandwidth isochronous endpoints.
6	ISO	R/W	0	ISO The CPU sets this bit to enable the receive endpoint for isochronous transfers, and clears it to enable the receive endpoint for bulk/interrupt transfers.

Bit/Field	Name	Type	Reset	Description
5	DMAEN	R/W	0	DMA Request Enable The CPU sets this bit to enable the DMA request for the receive endpoint.
4	DISNYET / PIDERR	R/W	0	Disable NYET / PID Error For bulk or interrupt transactions, the CPU sets this bit to disable the sending of NYET handshakes. When set, all successfully received packets are acknowledged, including at the point at which the FIFO becomes full. For ISO transactions, the core sets this bit to indicate a PID error in the received packet.
3	DMAMOD	R/W	0	DMA Request Mode The CPU sets this bit to select DMA Request Mode 1 and clears it to select DMA Request Mode 0.
2:1	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	INCRX	R/W0C	0	Incomplete Receive This bit is set in a high-bandwidth isochronous/interrupt transfer if the packet in the receive FIFO is incomplete because parts of the data were not received. It is cleared when RXRDY is cleared. **Note:** Only valid for isochronous transfers.

Register 68: USB Receive Byte Count Endpoint 1 (USBRXCOUNT1), offset 0x118

Register 69: USB Receive Byte Count Endpoint 2 (USBRXCOUNT2), offset 0x128

Register 70: USB Receive Byte Count Endpoint 3 (USBRXCOUNT3), offset 0x138

Host

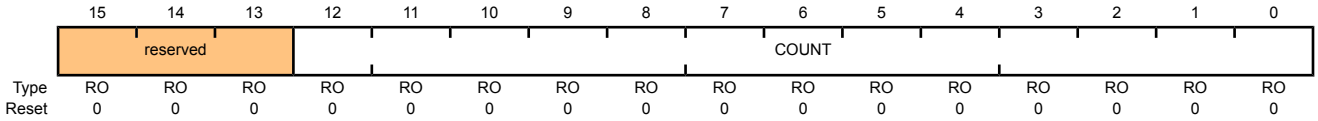
Note: The value returned changes as the FIFO is unloaded and is only valid while the RXRDY bit in the **USBRXCSRLn** register is set.

Device

USBRXCount1 is a 16-bit read-only register that holds the number of data bytes in the packet currently in line to be read from the receive FIFO. If the packet is transmitted as multiple bulk packets, the number given is for the combined packet.

USB Receive Byte Count Endpoint 1 (USBRXCOUNT1)

Base 0x4005.0000
Offset 0x118
Type RO, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:13	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12:0	COUNT	RO	0x00	Receive Packet Count Number of bytes in the receive packet.

Register 71: USB Host Transmit Configure Type Endpoint 1 (USBTXTYPE1), offset 0x11A

Register 72: USB Host Transmit Configure Type Endpoint 2 (USBTXTYPE2), offset 0x12A

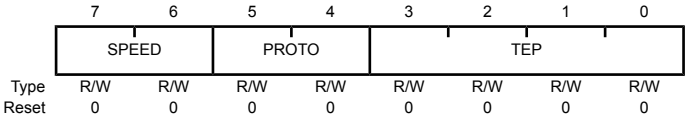
Register 73: USB Host Transmit Configure Type Endpoint 3 (USBTXTYPE3), offset 0x13A

Host

USBTXTYPE1 is an 8-bit register that should be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected transmit endpoint, and its operating speed.

USB Host Transmit Configure Type Endpoint 1 (USBTXTYPE1)

Base 0x4005.0000
Offset 0x11A
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:6	SPEED	R/W	0x00	Operating Speed Operating speed of the target device when the core is configured with the hub option: Value Description 00 Default The target is assumed to be using the same connection speed as the core. 01 Reserved 10 Full 11 Low When the core is not configured with the hub option, these bits should not be accessed
5:4	PROTO	R/W	0x00	Protocol The CPU should set this to select the required protocol for the transmit endpoint: Value Description 00 Control 01 Isochronous 10 Bulk 11 Interrupt

Bit/Field	Name	Type	Reset	Description
3:0	TEP	R/W	0x00	Target Endpoint Number The CPU should set this value to the endpoint number contained in the transmit endpoint descriptor returned to the USB controller during device enumeration.

Register 74: USB Host Transmit Interval Endpoint 1 (USBTXINTERVAL1), offset 0x11B

Register 75: USB Host Transmit Interval Endpoint 2 (USBTXINTERVAL2), offset 0x12B

Register 76: USB Host Transmit Interval Endpoint 3 (USBTXINTERVAL3), offset 0x13B

Host

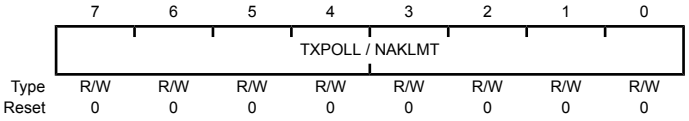
USBTXINTERVALn is an 8-bit register that, for interrupt and isochronous transfers, defines the polling interval for the currently selected transmit endpoint. For bulk endpoints, this register sets the number of frames after which the endpoint should time out on receiving a stream of NAK responses.

The **USBTXINTERVALn** register value defines a number of frames, as follows:

Transfer Type	Speed	Valid values (m)	Interpretation
Interrupt	Low-Speed or Full-Speed	1 – 255	Polling interval is *m* frames.
Isochronous	Full-Speed	1 – 16	Polling interval is $2^{(m-1)}$ frames.
Bulk	Full-Speed	2 – 16	NAK Limit is $2^{(m-1)}$ frames. A value of 0 or 1 disables the NAK timeout function.

USB Host Transmit Interval Endpoint 1 (USBTXINTERVAL1)

Base 0x4005.0000
Offset 0x11B
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:0	TXPOLL / NAKLMT	R/W	0x00	TX Polling / NAK Limit Polling interval for interrupt/isochronous transfers; NAK limit for bulk transfers.

Register 77: USB Host Configure Receive Type Endpoint 1 (USBRXTYPE1), offset 0x11C

Register 78: USB Host Configure Receive Type Endpoint 2 (USBRXTYPE2), offset 0x12C

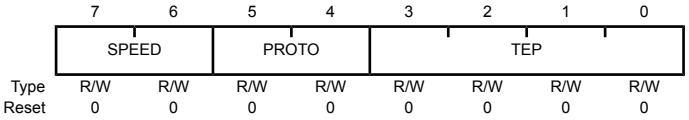
Register 79: USB Host Configure Receive Type Endpoint 3 (USBRXTYPE3), offset 0x13C

Host

USBRXTYPE1 is an 8-bit register that should be written with the endpoint number to be targeted by the endpoint, the transaction protocol to use for the currently selected receive endpoint, and its operating speed.

USB Host Configure Receive Type Endpoint 1 (USBRXTYPE1)

Base 0x4005.0000
Offset 0x11C
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:6	SPEED	R/W	0x00	Operating Speed Operating speed of the target device when the core is configured with the hub option. Value Description 00 Default The target is assumed to be using the same connection speed as the core. 01 Reserved 10 Full 11 Low When the core is not configured with the hub option, these bits should not be accessed.
5:4	PROTO	R/W	0x00	Protocol The CPU should set this to select the required protocol for the receive endpoint: Value Description 00 Control 01 Isochronous 10 Bulk 11 Interrupt

Bit/Field	Name	Type	Reset	Description
3:0	TEP	R/W	0x00	Target Endpoint Number The CPU should set this value to the endpoint number contained in the receive endpoint descriptor returned to the USB controller during device enumeration.

Register 80: USB Host Receive Polling Interval Endpoint 1 (USBRXINTERVAL1), offset 0x11D

Register 81: USB Host Receive Polling Interval Endpoint 2 (USBRXINTERVAL2), offset 0x12D

Register 82: USB Host Receive Polling Interval Endpoint 3 (USBRXINTERVAL3), offset 0x13D

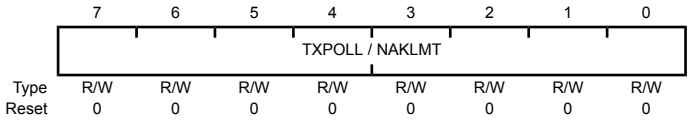
Host

USBRXINTERVAL1 is an 8-bit register that, for interrupt and isochronous transfers, defines the polling interval for the currently selected receive endpoint. For bulk endpoints, this register sets the number of frames after which the endpoint should time out on receiving a stream of NAK responses. The value that is set defines the number of frames, as follows:

Transfer Type	Speed	Valid Values (*m*)	Interpretation
Interrupt	Low-Speed or Full-Speed	1 – 255	Polling interval is *m* frames.
Isochronous	Full-Speed	1 – 16	Polling interval is $2^{(m-1)}$ frames.
Bulk	Full-Speed	2 – 16	NAK Limit is $2^{(m-1)}$ frames. **Note:** A value of 0 or 1 disables the NAK timeout function.

USB Host Receive Polling Interval Endpoint 1 (USBRXINTERVAL1)

Base 0x4005.0000
Offset 0x11D
Type R/W, reset 0x00



Bit/Field	Name	Type	Reset	Description
7:0	TXPOLL / NAKLMT	R/W	0x00	RX Polling / NAK Limit Polling interval for interrupt/isochronous transfers; NAK limit for bulk transfers.

Register 83: USB Request Packet Count in Block Transfer Endpoint 1 (USBRQPKTCOUNT1), offset 0x304

Register 84: USB Request Packet Count in Block Transfer Endpoint 2 (USBRQPKTCOUNT2), offset 0x308

Register 85: USB Request Packet Count in Block Transfer Endpoint 3 (USBRQPKTCOUNT3), offset 0x30C

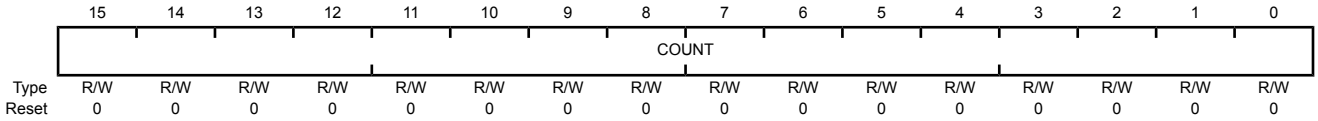
Host

This 16-bit read/write register is used in Host mode to specify the number of packets that are to be transferred in a block transfer of one or more bulk packets to receive endpoint *n*. The core uses the value recorded in this register to determine the number of requests to issue where the AUTORQ bit in the **USBRXCSRHn** register has been set. See “IN Transactions as a Host” on page 569.

Note: Multiple packets combined into a single bulk packet within the FIFO count as one packet.

USB Request Packet Count in Block Transfer Endpoint 1 (USBRQPKTCOUNT1)

Base 0x4005.0000
Offset 0x304
Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:0	COUNT	R/W	0x00	Block Transfer Packet Count Sets the number of packets of size MaxP that are to be transferred in a block transfer. **Note:** This is only used in Host mode when AUTORQ is set. The bit has no effect in Device mode or when AUTORQ is not set.

Register 86: USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS), offset 0x340

Host

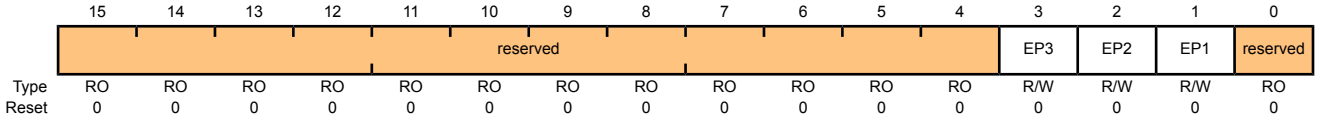
USBRXDPKTBUFDIS is a 16-bit register that indicates which of the receive endpoints have disabled the double-packet buffer functionality (see the section called “Double-Packet Buffering” on page 565).

Device

Note: Bits relating to endpoints that have not been configured may be asserted by writing a 1 to their respective register; however the disable bit will have no observable effect.

USB Receive Double Packet Buffer Disable (USBRXDPKTBUFDIS)

Base 0x4005.0000
Offset 0x340
Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EP3	R/W	0	EP3 RX Double-Packet Buffer Disable
2	EP2	R/W	0	EP2 RX Double-Packet Buffer Disable
1	EP1	R/W	0	EP1 RX Double-Packet Buffer Disable
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 87: USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS), offset 0x342

Host

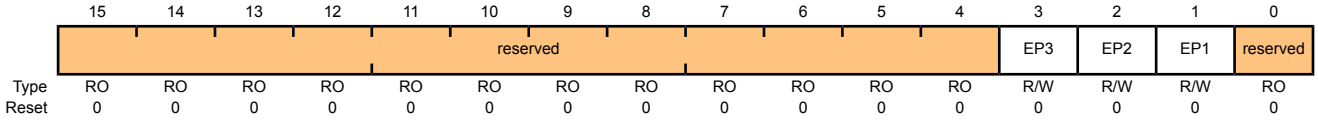
USBTXDPKTBUFDIS is a 16-bit register that indicates which of the transmit endpoints have disabled the double-packet buffer functionality (see the section called “Double-Packet Buffering” on page 564).

Device

Note: Bits relating to endpoints that have not been configured may be asserted by writing a 1 their respective register; however, the disable bit will have no observable effect.

USB Transmit Double Packet Buffer Disable (USBTXDPKTBUFDIS)

Base 0x4005.0000
Offset 0x342
Type R/W, reset 0x0000



Bit/Field	Name	Type	Reset	Description
15:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	EP3	R/W	0	EP3 TX Double-Packet Buffer Disable
2	EP2	R/W	0	EP2 TX Double-Packet Buffer Disable
1	EP1	R/W	0	EP1 TX Double-Packet Buffer Disable
0	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Register 88: USB External Power Control (USBEPC), offset 0x400

Host

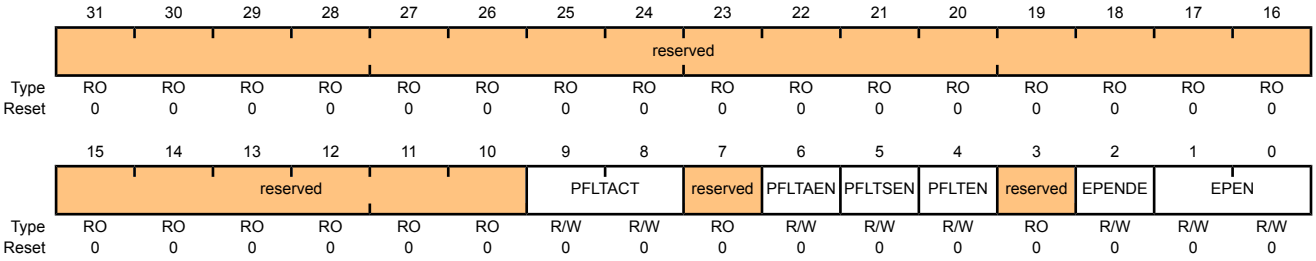
Device

OTG

USBEPC is instantiated in a USB unit in a wrapper around the USB controller/PHY IP. This 32-bit register specifies the function of the two-pin external power interface (USB0EPEN and USB0PFLT). The assertion of the power fault input may generate an automatic action, as controlled by the hardware configuration registers. The automatic action is necessary since the fault condition may require a response faster than one provided by firmware.

USB External Power Control (USBEPC)

Base 0x4005.0000
Offset 0x400
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
9:8	PFLTACT	R/W	0x00	Power Fault Action Specifies how the USB0EPEN signal is changed when detecting a USB power fault. Value Description 0x0 Unchanged USB0EPEN is controlled by the combination of the EPEN and EPENDE bits. 0x1 Tristate USB0EPEN is undriven (tristate). 0x2 Low USB0EPEN driven Low. 0x3 High USB0EPEN driven High.
7	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
6	PFLTAEN	R/W	0	Power Fault Action Enable Specifies whether a USB power fault triggers any automatic corrective action regarding the driven state of the USB0EPEN signal. Value Description 0 Disabled USB0EPEN is controlled by the combination of the EPEN and EPENDE bits. 1 Enabled The USB0EPEN output is automatically changed to the state as specified in the PFLTACT field.
5	PFLTSEN	R/W	0	Power Fault Sense Specifies the logical sense of the USB0PFLT input signal that indicates an error condition. The complementary state is the inactive state. Value Description 0 Low Fault If USB0PFLT is driven Low, the power fault is signaled internally (if enabled). 1 High Fault If USB0PFLT is driven High, the power fault is signaled internally (if enabled).
4	PFLTEN	R/W	0	Power Fault Input Enable Specifies whether the USB0PFLT input signal is used in internal logic. Value Description 0 Not Used The USB0PFLT signal is ignored. 1 Used The USB0PFLT signal is used internally.
3	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
2	EPENDE	R/W	0	EPEN Drive Enable Specifies whether the USB0EPEN signal is driven or undriven (tristate). When driven, the signal value is specified by the EPEN bit. When not driven, the EPEN bit is ignored and the USB0EPEN signal is placed in a high-impedance state. Value Description 0 Not Driven The USB0EPEN signal is high impedance. 1 Driven The USB0EPEN signal is driven to the logical value specified by the EPEN bit value. The USB0EPEN is undriven at reset since the sense of the external power supply enable is unknown. By adding high-impedance state, system designers may bias the power supply enable to the disabled state using a large resistor (100 kΩ) and later configure and drive the output signal to enable the power supply.
1:0	EPEN	R/W	0x00	External Power Supply Enable Configuration Specifies and controls the logical value driven on the USB0EPEN signal. Value Description 0x0 Power Enable Active Low The USB0EPEN signal is driven Low if EPENDE is 1. 0x1 Power Enable Active High The USB0EPEN signal is driven High if EPENDE is 1. 0x2-0x3 Reserved

Register 89: USB External Power Control Raw Interrupt Status (USBEPCRIS), offset 0x404

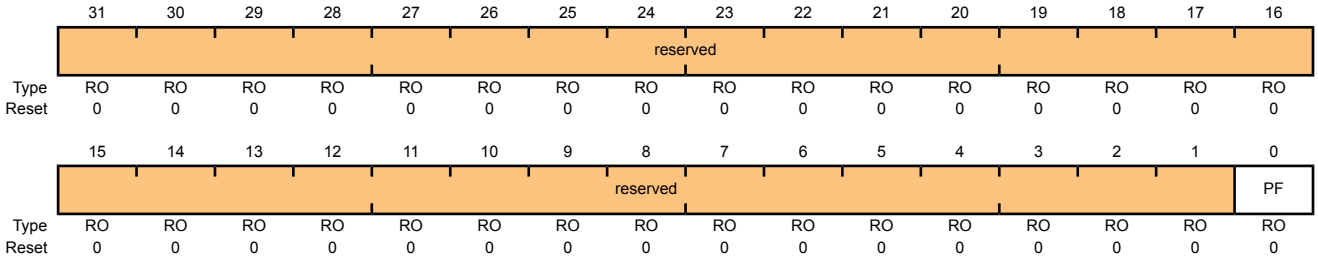
Host

USBEPCRIS is instantiated in a USB unit in a wrapper around the USB controller/PHY IP. This 32-bit register specifies the unmasked interrupt status of the two-pin external power interface.

Device

USB External Power Control Raw Interrupt Status (USBEPCRIS)

Base 0x4005.0000
Offset 0x404
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	PF	RO	0	USB Power Fault Interrupt Status Specifies the unmasked state of the power fault status. This bit is cleared by writing a 1 to the PF bit in the **USBEPCISC** register. Value Description 0 The hardware has not detected a power fault. 1 The hardware has detected a power fault.

Register 90: USB External Power Control Interrupt Mask (USBEPCIM), offset 0x408

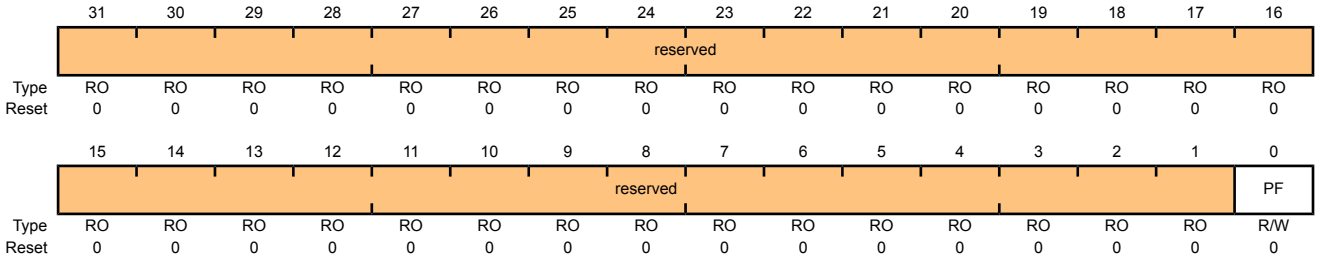
Host

USBEPCIM is instantiated in a USB unit in a wrapper around the USB controller/PHY IP. This 32-bit register specifies the interrupt mask of the two-pin external power interface.

Device

USB External Power Control Interrupt Mask (USBEPCIM)

Base 0x4005.0000
Offset 0x408
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	PF	R/W	0	USB Power Fault Interrupt Mask Specifies whether a detected power fault generates an interrupt. Value Description 0 No Interrupt The hardware does not generate an interrupt on detected power fault. 1 Interrupt The hardware generates an interrupt on detected power fault.

Register 91: USB External Power Control Interrupt Status and Clear (USBEPCISC), offset 0x40C

Host

Device

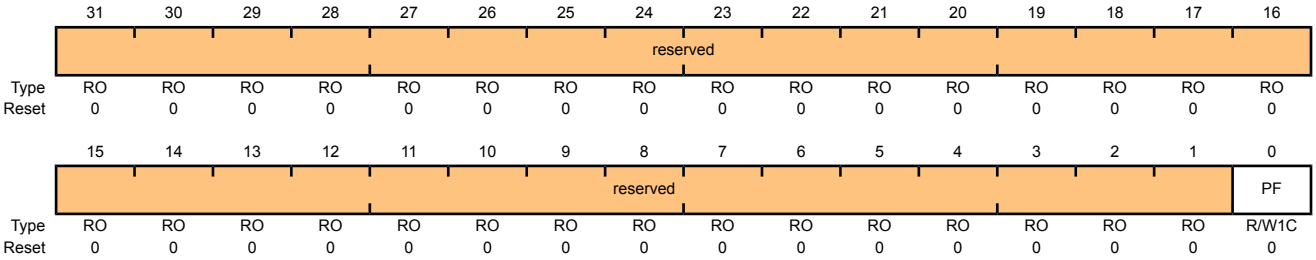
USBEPCISC is instantiated in a USB unit in a wrapper around the USB controller/PHY IP. This 32-bit register specifies the masked interrupt status of the two-pin external power interface. It also provides a method to clear the interrupt state.

USB External Power Control Interrupt Status and Clear (USBEPCISC)

Base 0x4005.0000

Offset 0x40C

Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	PF	R/W1C	0	USB Power Fault Interrupt Status and Clear Specifies whether a detected power fault has generated an interrupt. Value Description 0 No Interrupt The hardware has not generated an interrupt for a detected power fault condition. 1 Interrupt The hardware has generated an interrupt for a detected power fault condition. Writing a 1 to this bit clears it and the **USBEPCRIS** PF bit. This bit is set if the **USBEPCRIS** PF bit is set (by hardware) and the **USBEPCIM** PF bit is set.

Register 92: USB Device Resume Raw Interrupt Status (USBDRRIS), offset 0x410

Host

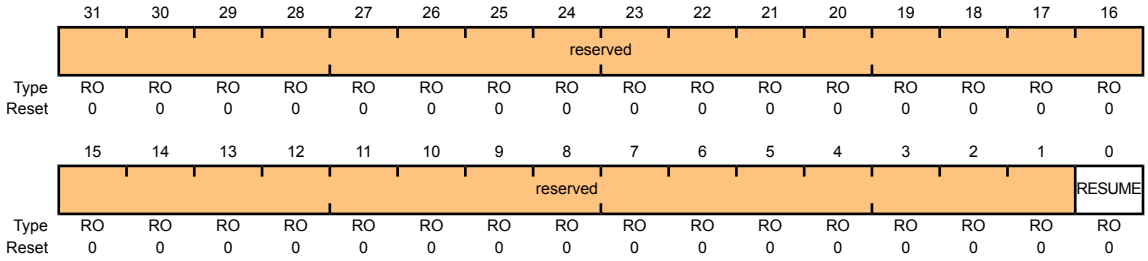
The **USBDRRIS** 32-bit register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

Device

USB Device Resume Raw Interrupt Status (USBDRRIS)

Base 0x4005.0000
Offset 0x410
Type RO, reset 0x0000.0000

OTG



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	RESUME	RO	0	Resume Interrupt Status Specifies the unmasked state of the resume status. This bit is cleared by writing a 1 to the RESUME bit in the **USBDRISC** register. Value Description 0 The hardware has not detected a Resume. 1 The hardware has detected a Resume.

Register 93: USB Device Resume Interrupt Mask (USBDRIM), offset 0x414

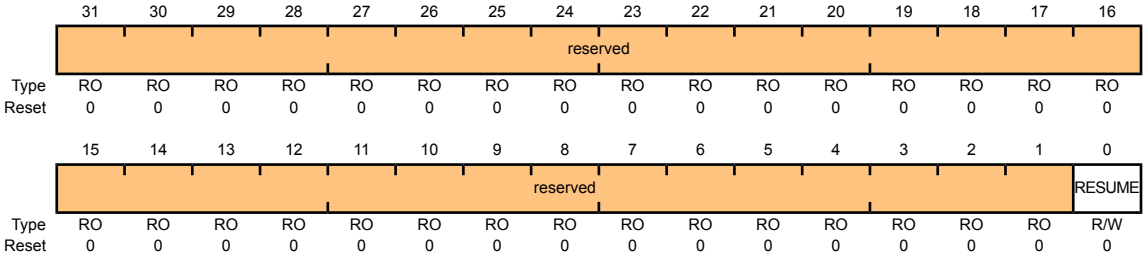
Host

Device

OTG

The **USBDRIM** 32-bit register is the masked interrupt status register. On a read, this register gives the current masked status value of the corresponding interrupt. A write has no effect.

USB Device Resume Interrupt Mask (USBDRIM)
Base 0x4005.0000
Offset 0x414
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	RESUME	R/W	0	Resume Interrupt Mask Specifies whether a detected Resume generates an interrupt. Value Description 0 No Interrupt The hardware does not generate an interrupt on detected Resume. 1 Interrupt The hardware generates an interrupt on detected Resume. This should only be enabled when a suspend has been detected (Suspend bit in **USBIS** register).

Register 94: USB Device Resume Interrupt Status and Clear (USBDRISC), offset 0x418

Host

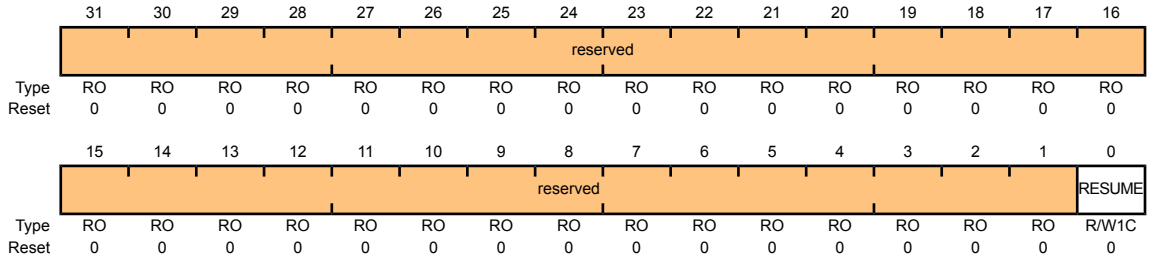
The **USBDRISC** 32-bit register is the interrupt clear register. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

Device

USB Device Resume Interrupt Status and Clear (USBDRISC)

Base 0x4005.0000
Offset 0x418
Type W1C, reset 0x0000.0000

OTG



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	RESUME	R/W1C	0	Resume Interrupt Status and Clear Specifies whether a detected Resume has generated an interrupt. Value Description 0 No Interrupt The hardware has not generated an interrupt for a detected Resume. 1 Interrupt The hardware has generated an interrupt for a detected Resume. Writing a 1 to this bit clears it and the **USBDRRIS** RESUME bit. This bit is set if the **USBDRRIS** RESUME bit is set (by hardware) and the **USBEDRIM** RESUME bit is set.

Register 95: USB General-Purpose Control and Status (USBGPCS), offset 0x41C

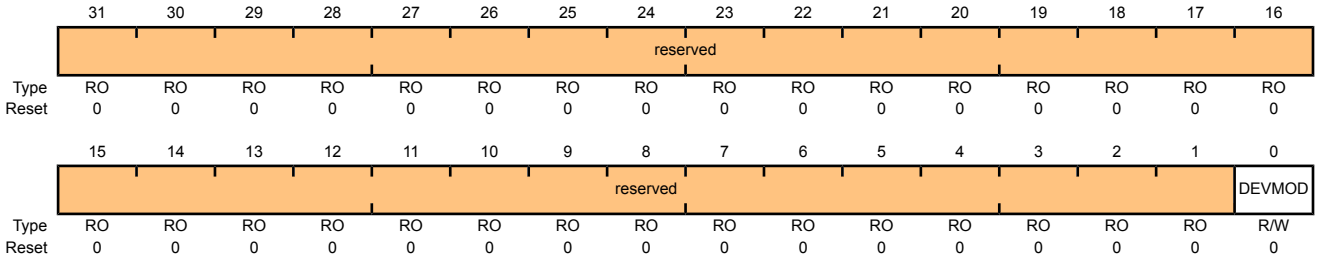
Host

USBGPCS provides the state of the internal ID signal. Set to force to Device mode. Clear to force to Host mode.

Device

USB General-Purpose Control and Status (USBGPCS)

Base 0x4005.0000
Offset 0x41C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	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	DEVMOD	R/W	0	Device Mode This bit is used to control the state of the internal ID signal. In Device mode this bit is ignored (assumed set). Value Description 0 Host mode 1 Device mode

19 Analog Comparators

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

Note: Not all comparators have the option to drive an output pin.

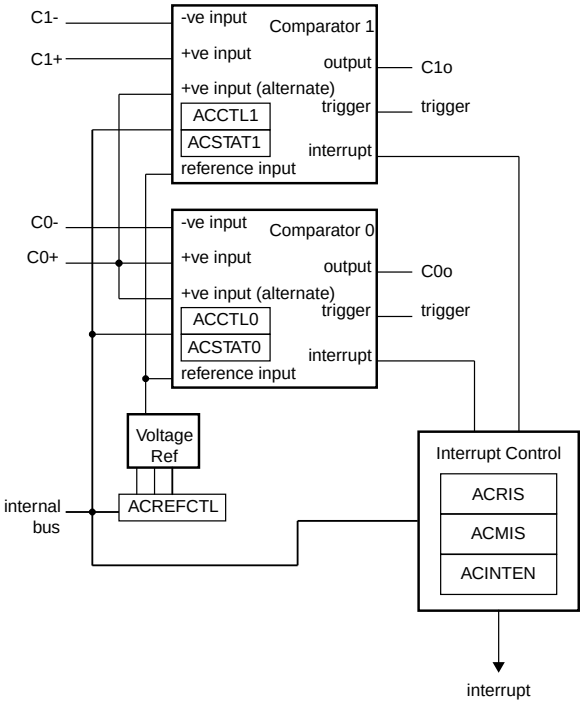
The comparator can provide its output to a device pin, acting as a replacement for an analog comparator on the board, or it can be used to signal the application via interrupts or triggers to the ADC to cause it to start capturing a sample sequence. The interrupt generation and ADC triggering logic is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

The Stellaris® Analog Comparators module has the following features:

- Two independent integrated analog comparators
- Configurable for output to drive an output pin, generate an interrupt, or initiate an ADC sample sequence
- Compare external pin input to external pin input or to internal programmable voltage reference
- 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

19.1 Block Diagram

Figure 19-1. Analog Comparator Module Block Diagram



19.2 Functional Description

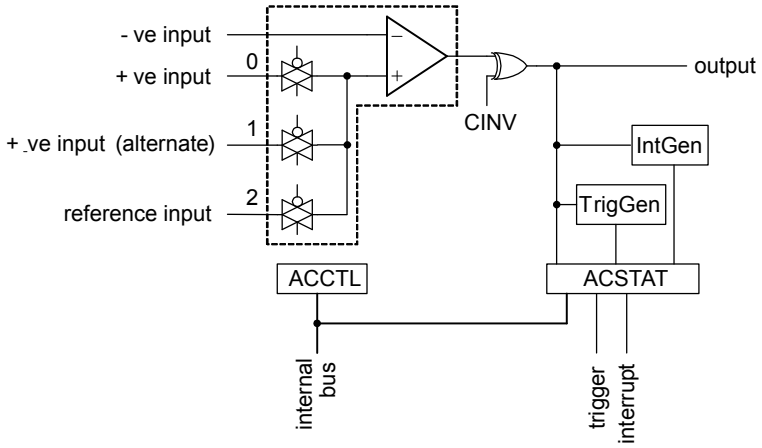
Important: It is recommended that the Digital-Input enable (the GPIODEN bit in the GPIO module) for the analog input pin be disabled to prevent excessive current draw from the I/O pads.

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 19-2 on page 650, 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 19-2. Structure of Comparator Unit



A comparator is configured through two status/control registers (**ACCTL** and **ACSTAT**). The internal reference is configured through one control register (**ACREFCTL**). Interrupt status and control is configured through three registers (**ACMIS**, **ACRIS**, and **ACINTEN**).

Typically, the comparator output is used internally to generate controller interrupts. It may also be used to drive an external pin or generate an analog-to-digital converter (ADC) trigger.

19.2.1 Internal Reference Programming

The structure of the internal reference is shown in Figure 19-3 on page 650. This is controlled by a single configuration register (**ACREFCTL**). Table 19-1 on page 650 shows the programming options to develop specific internal reference values, to compare an external voltage against a particular voltage generated internally.

Figure 19-3. Comparator Internal Reference Structure

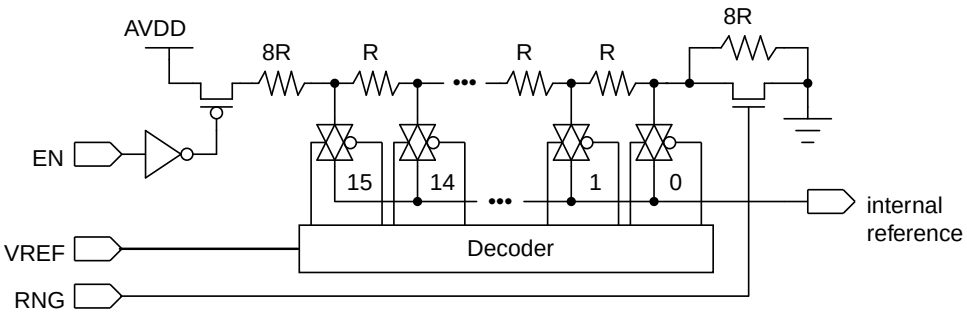


Table 19-1. Internal Reference Voltage and ACREFCTL Field Values

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 31 R. $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF + 8)}{31}$ $V_{REF} = 0.85 + 0.106 \times VREF$ The range of internal reference in this mode is 0.85-2.448 V.
	RNG=1	Total resistance in ladder is 23 R. $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{VREF}{23}$ $V_{REF} = 0.143 \times VREF$ The range of internal reference for this mode is 0-2.152 V.

19.3 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 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.

19.4 Register Map

Table 19-2 on page 652 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 19-2. Analog Comparators Register Map

Offset	Name	Type	Reset	Description	See page
0x000	ACMIS	R/W1C	0x0000.0000	Analog Comparator Masked Interrupt Status	653
0x004	ACRIS	RO	0x0000.0000	Analog Comparator Raw Interrupt Status	654
0x008	ACINTEN	R/W	0x0000.0000	Analog Comparator Interrupt Enable	655
0x010	ACREFCTL	R/W	0x0000.0000	Analog Comparator Reference Voltage Control	656
0x020	ACSTAT0	RO	0x0000.0000	Analog Comparator Status 0	657
0x024	ACCTL0	R/W	0x0000.0000	Analog Comparator Control 0	658
0x040	ACSTAT1	RO	0x0000.0000	Analog Comparator Status 1	657
0x044	ACCTL1	R/W	0x0000.0000	Analog Comparator Control 1	658

19.5 Register Descriptions

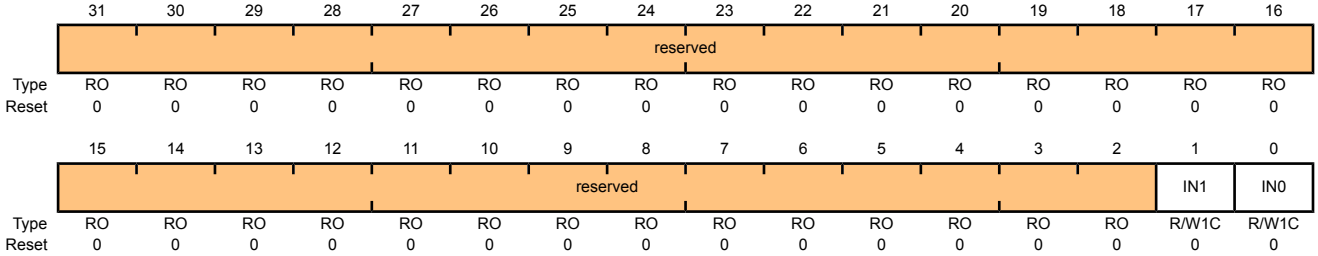
The remainder of this section lists and describes the Analog Comparator registers, in numerical order by address offset.

Register 1: Analog Comparator Masked Interrupt Status (ACMIS), offset 0x000

This register provides a summary of the interrupt status (masked) of the comparators.

Analog Comparator Masked Interrupt Status (ACMIS)

Base 0x4003.C000
Offset 0x000
Type R/W1C, reset 0x0000.0000



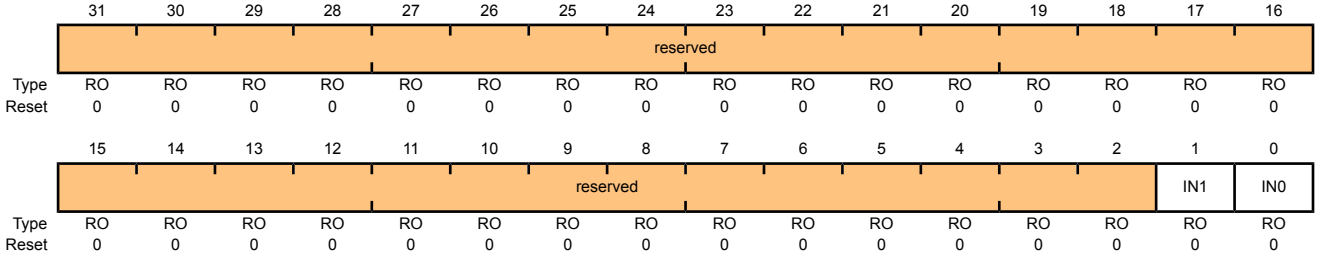
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	IN1	R/W1C	0	Comparator 1 Masked Interrupt Status Gives the masked interrupt state of this interrupt. Write 1 to this bit to clear the pending interrupt.
0	IN0	R/W1C	0	Comparator 0 Masked Interrupt Status Gives the masked interrupt state of this interrupt. Write 1 to this bit to clear the pending interrupt.

Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x004

This register provides a summary of the interrupt status (raw) of the comparators.

Analog Comparator Raw Interrupt Status (ACRIS)

Base 0x4003.C000
Offset 0x004
Type RO, reset 0x0000.0000



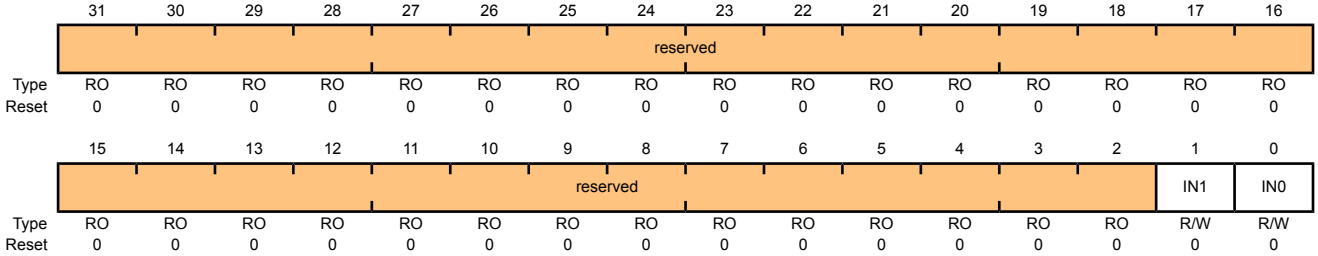
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	IN1	RO	0	Comparator 1 Interrupt Status When set, indicates that an interrupt has been generated by comparator 1.
0	IN0	RO	0	Comparator 0 Interrupt Status When set, indicates that an interrupt has been generated by comparator 0.

Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x008

This register provides the interrupt enable for the comparators.

Analog Comparator Interrupt Enable (ACINTEN)

Base 0x4003.C000
Offset 0x008
Type R/W, reset 0x0000.0000



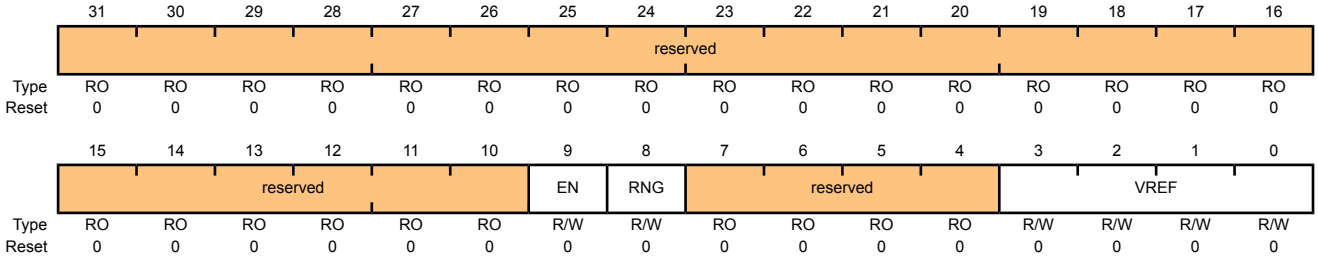
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	IN1	R/W	0	Comparator 1 Interrupt Enable When set, enables the controller interrupt from the comparator 1 output.
0	IN0	R/W	0	Comparator 0 Interrupt Enable When set, enables the controller interrupt from the comparator 0 output.

Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x010

This register specifies whether the resistor ladder is powered on as well as the range and tap.

Analog Comparator Reference Voltage Control (ACREFCTL)

Base 0x4003.C000
Offset 0x010
Type R/W, reset 0x0000.0000



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 31 R. If 1, the resistor ladder has a total resistance of 23 R.
7:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3:0	VREF	R/W	0x00	Resistor Ladder Voltage Ref The 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 19-1 on page 650 for some output reference voltage examples.

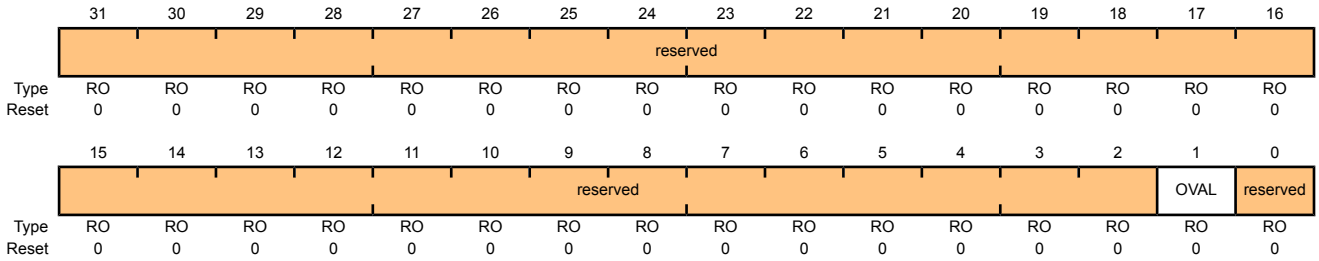
Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x020

Register 6: Analog Comparator Status 1 (ACSTAT1), offset 0x040

These registers specify the current output value of the comparator.

Analog Comparator Status 0 (ACSTAT0)

Base 0x4003.C000
Offset 0x020
Type RO, reset 0x0000.0000



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.

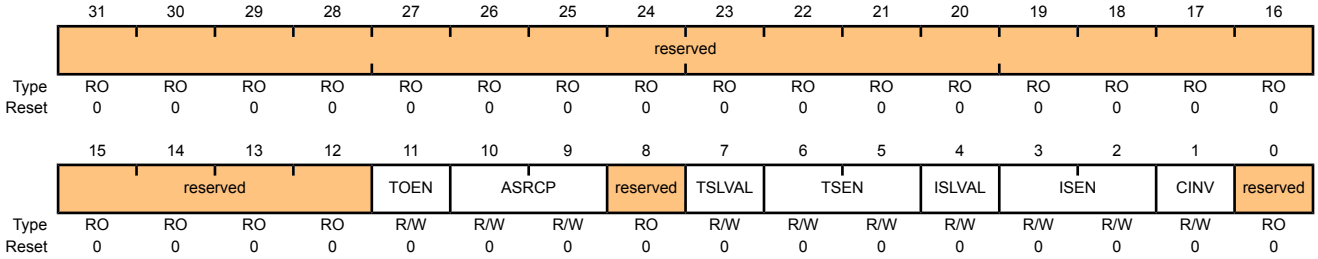
Register 7: Analog Comparator Control 0 (ACCTL0), offset 0x024

Register 8: Analog Comparator Control 1 (ACCTL1), offset 0x044

These registers configure the comparator's input and output.

Analog Comparator Control 0 (ACCTL0)

Base 0x4003.C000
Offset 0x024
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11	TOEN	R/W	0	Trigger Output Enable The TOEN bit enables the ADC event transmission to the ADC. If 0, the event is suppressed and not sent to the ADC. If 1, the event is transmitted to the ADC.
10:9	ASRCP	R/W	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	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	TSLVAL	R/W	0	Trigger Sense Level Value The TSLVAL bit specifies the sense value of the input that generates an ADC event if in Level Sense mode. If 0, an ADC event is generated if the comparator output is Low. Otherwise, an ADC event is generated if the comparator output is High.

Bit/Field	Name	Type	Reset	Description
6:5	TSEN	R/W	0x0	Trigger Sense The TSEN field specifies the sense of the comparator output that generates an ADC event. The sense conditioning is as follows: Value Function 0x0 Level sense, see TSLVAL 0x1 Falling edge 0x2 Rising edge 0x3 Either edge
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.
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.

20 Pulse Width Modulator (PWM)

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

The Stellaris® PWM module consists of four PWM generator blocks and a control block. The control block determines the polarity of the PWM signals, and which signals are passed through to the pins.

Each PWM generator block produces two PWM signals that can either be independent signals (other than being based on the same timer and therefore having the same frequency) or a single pair of complementary signals with dead-band delays inserted. The output of the PWM generation blocks are managed by the output control block before being passed to the device pins.

The Stellaris® PWM module provides a great deal of flexibility. It can generate simple PWM signals, such as those required by a simple charge pump. It can also generate paired PWM signals with dead-band delays, such as those required by a half-H bridge driver. Three generator blocks can also generate the full six channels of gate controls required by a 3-phase inverter bridge.

Each Stellaris® PWM module has the following features:

- Four PWM generator blocks, each with one 16-bit counter, two PWM comparators, a PWM signal generator, a dead-band generator, and an interrupt/ADC-trigger selector
- Four fault inputs in hardware to promote low-latency shutdown
- One 16-bit counter
  - Runs in Down or Up/Down mode
  - Output frequency controlled by a 16-bit load value
  - Load value updates can be synchronized
  - Produces output signals at zero and load value
- Two PWM comparators
  - Comparator value updates can be synchronized
  - Produces output signals on match
- PWM generator
  - Output PWM signal is constructed based on actions taken as a result of the counter and PWM comparator output signals
  - Produces two independent PWM signals
- Dead-band generator
  - Produces two PWM signals with programmable dead-band delays suitable for driving a half-H bridge
  - Can be bypassed, leaving input PWM signals unmodified

- Flexible output control block with PWM output enable of each PWM signal
  - PWM output enable of each PWM signal
  - Optional output inversion of each PWM signal (polarity control)
  - Optional fault handling for each PWM signal
  - Synchronization of timers in the PWM generator blocks
  - Synchronization of timer/comparator updates across the PWM generator blocks
  - Interrupt status summary of the PWM generator blocks
- Can initiate an ADC sample sequence

20.1 Block Diagram

Figure 20-1 on page 661 provides the Stellaris® PWM module unit diagram and Figure 20-2 on page 662 provides a more detailed diagram of a Stellaris® PWM generator. The LM3S5749 controller contains four generator blocks (PWM0, PWM1, PWM2, and PWM3) and generates eight independent PWM signals or four paired PWM signals with dead-band delays inserted.

Figure 20-1. PWM Unit Diagram

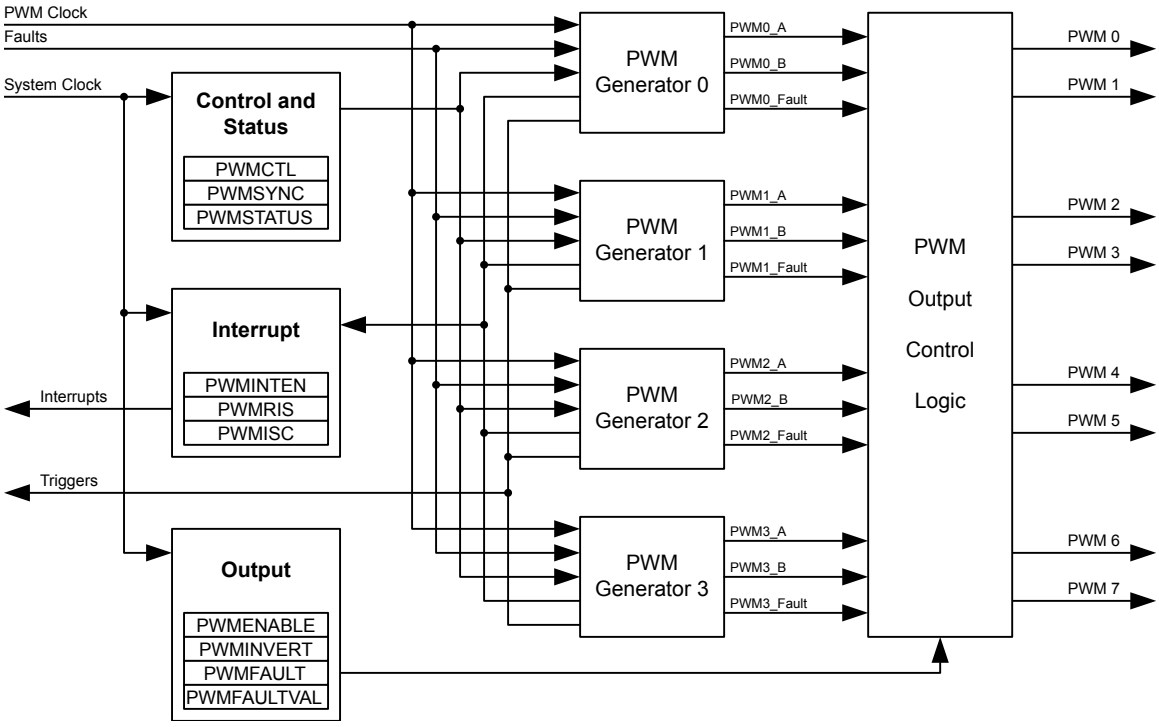
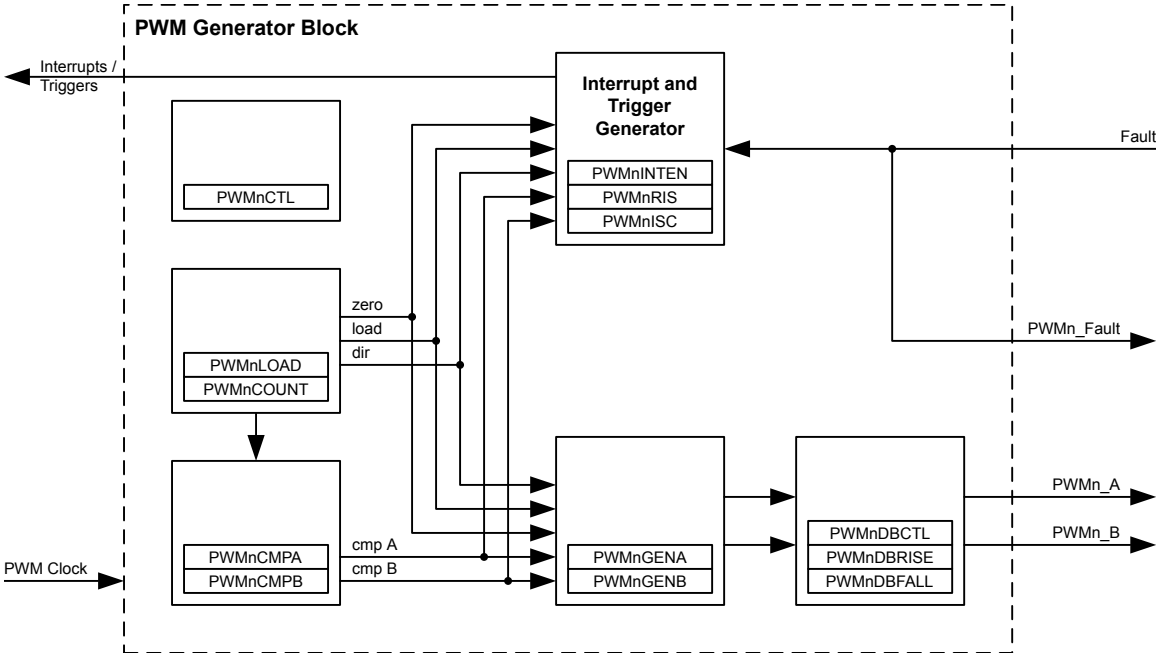


Figure 20-2. PWM Module Block Diagram



20.2 Functional Description

20.2.1 PWM Timer

The timer in each PWM generator runs in one of two modes: Count-Down mode or Count-Up/Down mode. In Count-Down mode, the timer counts from the load value to zero, goes back to the load value, and continues counting down. In Count-Up/Down mode, the timer counts from zero up to the load value, back down to zero, back up to the load value, and so on. Generally, Count-Down mode is used for generating left- or right-aligned PWM signals, while the Count-Up/Down mode is used for generating center-aligned PWM signals.

The timers output three signals that are used in the PWM generation process: the direction signal (this is always Low in Count-Down mode, but alternates between Low and High in Count-Up/Down mode), a single-clock-cycle-width High pulse when the counter is zero, and a single-clock-cycle-width High pulse when the counter is equal to the load value. Note that in Count-Down mode, the zero pulse is immediately followed by the load pulse.

20.2.2 PWM Comparators

There are two comparators in each PWM generator that monitor the value of the counter; when either match the counter, they output a single-clock-cycle-width High pulse. When in Count-Up/Down mode, these comparators match both when counting up and when counting down; they are therefore qualified by the counter direction signal. These qualified pulses are used in the PWM generation process. If either comparator match value is greater than the counter load value, then that comparator never outputs a High pulse.

Figure 20-3 on page 663 shows the behavior of the counter and the relationship of these pulses when the counter is in Count-Down mode. Figure 20-4 on page 663 shows the behavior of the counter and the relationship of these pulses when the counter is in Count-Up/Down mode.

Figure 20-3. PWM Count-Down Mode

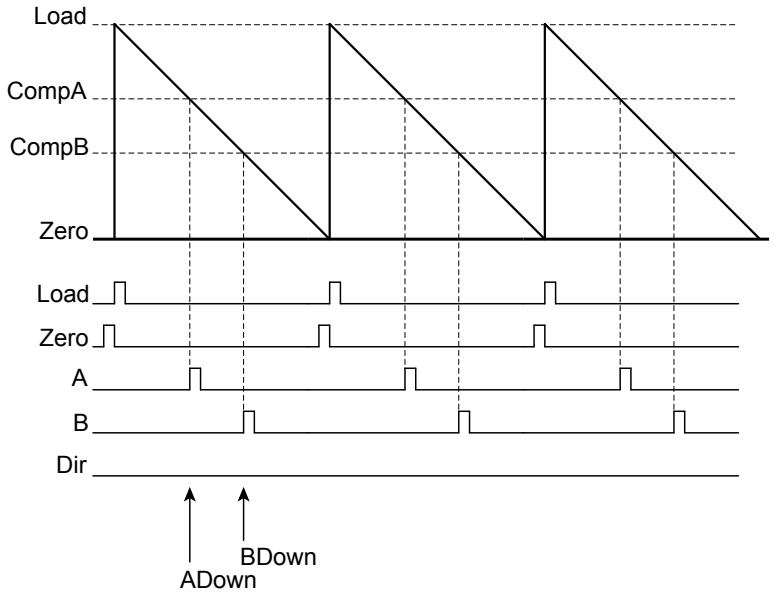
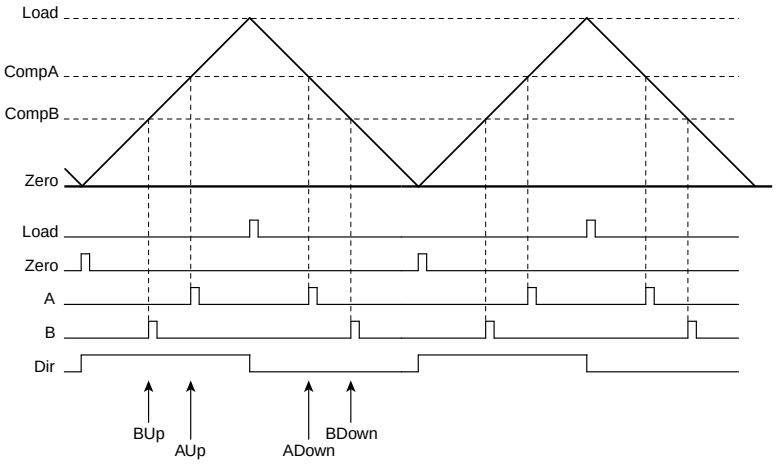


Figure 20-4. PWM Count-Up/Down Mode

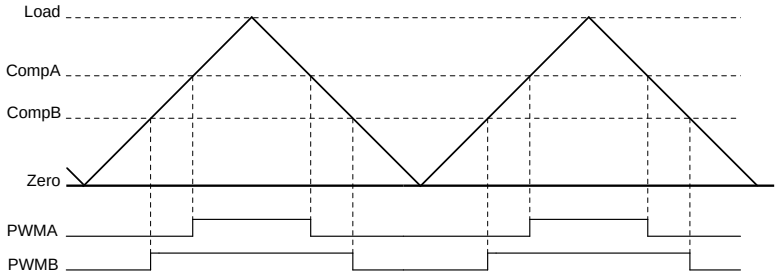


20.2.3 PWM Signal Generator

The PWM generator takes these pulses (qualified by the direction signal), and generates two PWM signals. In Count-Down mode, there are four events that can affect the PWM signal: zero, load, match A down, and match B down. In Count-Up/Down mode, there are six events that can affect the PWM signal: zero, load, match A down, match A up, match B down, and match B up. The match A or match B events are ignored when they coincide with the zero or load events. If the match A and match B events coincide, the first signal, PWMA, is generated based only on the match A event, and the second signal, PWMB, is generated based only on the match B event.

For each event, the effect on each output PWM signal is programmable: it can be left alone (ignoring the event), it can be toggled, it can be driven Low, or it can be driven High. These actions can be used to generate a pair of PWM signals of various positions and duty cycles, which do or do not overlap. Figure 20-5 on page 664 shows the use of Count-Up/Down mode to generate a pair of center-aligned, overlapped PWM signals that have different duty cycles.

Figure 20-5. PWM Generation Example In Count-Up/Down Mode



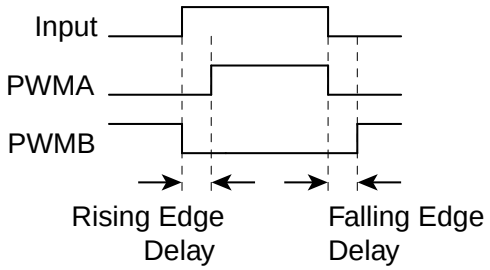
In this example, the first generator is set to drive High on match A up, drive Low on match A down, and ignore the other four events. The second generator is set to drive High on match B up, drive Low on match B down, and ignore the other four events. Changing the value of comparator A changes the duty cycle of the PWMA signal, and changing the value of comparator B changes the duty cycle of the PWMB signal.

20.2.4 Dead-Band Generator

The two PWM signals produced by the PWM generator are passed to the dead-band generator. If disabled, the PWM signals simply pass through unmodified. If enabled, the second PWM signal is lost and two PWM signals are generated based on the first PWM signal. The first output PWM signal is the input signal with the rising edge delayed by a programmable amount. The second output PWM signal is the inversion of the input signal with a programmable delay added between the falling edge of the input signal and the rising edge of this new signal.

This is therefore a pair of active High signals where one is always High, except for a programmable amount of time at transitions where both are Low. These signals are therefore suitable for driving a half-H bridge, with the dead-band delays preventing shoot-through current from damaging the power electronics. Figure 20-6 on page 664 shows the effect of the dead-band generator on an input PWM signal.

Figure 20-6. PWM Dead-Band Generator



20.2.5 Interrupt/ADC-Trigger Selector

The PWM generator also takes the same four (or six) counter events and uses them to generate an interrupt or an ADC trigger. Any of these events or a set of these events can be selected as a source for an interrupt; when any of the selected events occur, an interrupt is generated. Additionally, the same event, a different event, the same set of events, or a different set of events can be selected as a source for an ADC trigger; when any of these selected events occur, an ADC trigger pulse is generated. The selection of events allows the interrupt or ADC trigger to occur at a specific position within the PWM signal. Note that interrupts and ADC triggers are based on the raw events; delays in the PWM signal edges caused by the dead-band generator are not taken into account.

20.2.6 Synchronization Methods

The PWM unit provides four PWM generators providing eight PWM outputs that may be used in a wide variety of applications. Generally speaking, this falls into combinations of two categories of operation:

- **Unsynchronized.** The PWM generator and its two output signals are used by itself, independent of other PWM generators.
- **Synchronized.** The PWM generator and its two outputs signals are used in conjunction with other PWM generators using a common, unified time base.

If multiple PWM generators are configured with the same counter load value, this can be used to guarantee that they also have the same count value (this does imply that the PWM generators must be configured before they are synchronized). With this, more than two PWM signals can be produced with a known relationship between the edges of those signals since the counters always have the same values. Other states in the unit provide mechanisms to maintain the common time base and mutual synchronization.

The counter in a PWM unit generator can be reset to zero by writing the **PWM Time Base Sync (PWMSYNC)** register and setting the `Sync` bit associated with the generator. Multiple PWM generators can be synchronized together by setting all necessary `Sync` bits in one access. For example, setting the `Sync0` and `Sync1` bits in the **PWMSYNC** register causes the counters in PWM generators 0 and 1 to reset together.

Additionally, the state of a PWM unit is affected by writing to the registers of the PWM unit and the PWM units' generators, which has an effect on the synchronization between multiple PWM generators. Depending on the register accessed, the register state is updated in one of the following three ways:

- **Immediately.** The write value has immediate effect, and the hardware reacts immediately.
- **Locally Synchronized.** The write value does not affect the logic until the counter reaches the value zero. In this case, the effect of the write is deferred until the end of the PWM cycle (when the counter reaches zero). By waiting for the counter to reach zero, a guaranteed behavior is defined, and overly short or overly long output PWM pulses are prevented.
- **Globally Synchronized.** The write value does not affect the logic until two sequential events have occurred: (1) the global synchronization bit applicable to the generator is set, and (2) the counter reaches zero. In this case, the effect of the write is deferred until the end of the PWM cycle (when the counter reaches zero) following the end of all updates. This mode allows multiple items in multiple PWM generators to be updated simultaneously without odd effects during the update; everything runs from the old values until a point at which they all run from the new values. The Update mode of the load and comparator match values can be individually configured in each PWM generator block. It typically makes sense to use the synchronous update mechanism across PWM generator blocks when the timers in those blocks are synchronized, although this is not required in order for this mechanism to function properly.

The following registers provide either local or global synchronization based on the state of the **PWMnCTL** register `Update` bit value:

- Generator Registers: **PWMnLOAD**, **PWMnCMPA**, and **PWMnCMPB**

The following registers are provided with the optional functionality of synchronously updating rather than having all updates take immediate effect. The default update mode is immediate.

- Module-Level Register: **PWMENABLE**
- Generator Register: **PWMnGENA**, **PWMnGENB**, **PWMnDBCTL**, **PWMnDBRISE**, and **PWMnDBFALL**.

All other registers are considered statically provisioned for the execution of an application or are used dynamically for purposes unrelated to maintaining synchronization, and therefore, do not need synchronous update functionality.

20.2.7 Fault Conditions

A fault condition is one in which the controller must be signaled to stop normal PWM function and then sets the outputs to a safe state. There are two basic situations where this becomes necessary:

- The controller is stalled and cannot perform the necessary computation in the time required for motion control
- An external error or event is detected, such as an error

The PWM unit can use the following inputs to generate a fault condition, including:

- FAULTn pin assertion
- A stall of the controller generated by the debugger

Fault conditions are calculated on a per-PWM generator basis. Each PWM generator configures the necessary conditions to indicate a fault condition exists. This method allows the development of applications with dependent and independent control.

Each PWM generator's mode control, including fault condition handling, is provided in the **PWMnCTL** register. This register determines whether a single FAULT0 input is used (as previous Stellaris® products support) or whether all FAULTn input signals may be used to generate a fault condition. This register allows the fault condition duration to last as long as the external condition lasts, or it may specify that the external condition be latched and the fault condition (and its effects) last until cleared by software. Finally, this register also enables a counter that may be used to extend the period of a fault condition for external events to assure that the duration is a minimum length. The minimum fault period count is specified in the **PWMnMINFLTPER** register.

These PWM generator registers provide status, control, and configure the fault condition in each PWM generator: **PWMnFLTSRC0**, **PWMnFLTSTAT0**, and **PWMnFLTSEN**.

There are up to four FAULT input pins (FAULT0-FAULT3). These pins may be used with circuits that generate an active High or active Low signal to indicate an error condition. Each of the FAULTn pins may be individually programmed for this logic sense using the **PWMnFLTSEN** register.

The **PWMnFLTSRC0** register define the contribution of the external fault sources. Using these registers, individual or groups of FAULTn signals are ORed together to specify the external fault generating conditions.

Status regarding the specific fault cause is provided in **PWMnFLTSTAT0**.

PWM generator fault conditions may be promoted to a controller interrupt using the **PWMINTEN** register.

During fault conditions, the PWM output signals usually require being driven to safe values so that external equipment may be safely controlled. To facilitate this, the **PWMFAULT** register is used to

determine if the generated signal continues to be passed driven, or a specific fault condition encoding is driven on the PWM output, as specified in the **PWMFAULTVAL** register.

20.2.8 Output Control Block

With each PWM generator block producing two raw PWM signals, the output control block takes care of the final conditioning of the PWM signals before they go to the pins. Via a single register, the set of PWM signals that are actually enabled to the pins can be modified; this can be used, for example, to perform commutation of a brushless DC motor with a single register write (and without modifying the individual PWM generators, which are modified by the feedback control loop). Similarly, fault control can disable any of the PWM signals as well. A final inversion can be applied to any of the PWM signals, making them active Low instead of the default active High.

20.3 Initialization and Configuration

The following example shows how to initialize the PWM Generator 0 with a 25-KHz frequency, and with a 25% duty cycle on the PWM0 pin and a 75% duty cycle on the PWM1 pin. This example assumes the system clock is 20 MHz.

1. Enable the PWM clock by writing a value of 0x0010.0000 to the **RCGC0** register in the System Control module.
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.
4. Configure the **Run-Mode Clock Configuration (RCC)** register in the System Control module to use the PWM divide (USEPWMDIV) and set the divider (PWMDIV) to divide by 2 (000).
5. Configure the PWM generator for countdown mode with immediate updates to the parameters.
   - Write the **PWM0CTL** register with a value of 0x0000.0000.
   - Write the **PWM0GENA** register with a value of 0x0000.008C.
   - Write the **PWM0GENB** register with a value of 0x0000.080C.
6. Set the period. For a 25-KHz frequency, the period = 1/25,000, or 40 microseconds. The PWM clock source is 10 MHz; the system clock divided by 2. This translates to 400 clock ticks per period. Use this value to set the **PWM0LOAD** register. In Count-Down mode, set the Load field in the **PWM0LOAD** register to the requested period minus one.
   - Write the **PWM0LOAD** register with a value of 0x0000.018F.
7. Set the pulse width of the PWM0 pin for a 25% duty cycle.
   - Write the **PWM0CMPA** register with a value of 0x0000.012B.
8. Set the pulse width of the PWM1 pin for a 75% duty cycle.
   - Write the **PWM0CMPB** register with a value of 0x0000.0063.
9. Start the timers in PWM generator 0.

- Write the **PWM0CTL** register with a value of 0x0000.0001.

10. Enable PWM outputs.

- Write the **PWMENABLE** register with a value of 0x0000.0003.

20.4 Register Map

Table 20-1 on page 668 lists the PWM registers. The offset listed is a hexadecimal increment to the register's address, relative to the PWM base address of 0x4002.8000.

Table 20-1. PWM Register Map

Offset	Name	Type	Reset	Description	See page
0x000	PWMCTL	R/W	0x0000.0000	PWM Master Control	671
0x004	PWMSYNC	R/W	0x0000.0000	PWM Time Base Sync	672
0x008	PWMENABLE	R/W	0x0000.0000	PWM Output Enable	673
0x00C	PWMINVERT	R/W	0x0000.0000	PWM Output Inversion	675
0x010	PWMFAULT	R/W	0x0000.0000	PWM Output Fault	676
0x014	PWMINTEN	R/W	0x0000.0000	PWM Interrupt Enable	678
0x018	PWMRIS	RO	0x0000.0000	PWM Raw Interrupt Status	680
0x01C	PWMISC	R/W1C	0x0000.0000	PWM Interrupt Status and Clear	682
0x020	PWMSTATUS	RO	0x0000.0000	PWM Status	684
0x024	PWMFAULTVAL	R/W	0x0000.0000	PWM Fault Condition Value	685
0x040	PWM0CTL	R/W	0x0000.0000	PWM0 Control	687
0x044	PWM0INTEN	R/W	0x0000.0000	PWM0 Interrupt and Trigger Enable	692
0x048	PWM0RIS	RO	0x0000.0000	PWM0 Raw Interrupt Status	694
0x04C	PWM0ISC	R/W1C	0x0000.0000	PWM0 Interrupt Status and Clear	695
0x050	PWM0LOAD	R/W	0x0000.0000	PWM0 Load	696
0x054	PWM0COUNT	RO	0x0000.0000	PWM0 Counter	697
0x058	PWM0CMPA	R/W	0x0000.0000	PWM0 Compare A	698
0x05C	PWM0CMPB	R/W	0x0000.0000	PWM0 Compare B	699
0x060	PWM0GENA	R/W	0x0000.0000	PWM0 Generator A Control	700
0x064	PWM0GENB	R/W	0x0000.0000	PWM0 Generator B Control	703
0x068	PWM0DBCTL	R/W	0x0000.0000	PWM0 Dead-Band Control	706
0x06C	PWM0DBRISE	R/W	0x0000.0000	PWM0 Dead-Band Rising-Edge Delay	707
0x070	PWM0DBFALL	R/W	0x0000.0000	PWM0 Dead-Band Falling-Edge-Delay	708
0x074	PWM0FLTSRC0	R/W	0x0000.0000	PWM0 Fault Source 0	709
0x07C	PWM0MINFLTPER	R/W	0x0000.0000	PWM0 Minimum Fault Period	711

Offset	Name	Type	Reset	Description	See page
0x080	PWM1CTL	R/W	0x0000.0000	PWM1 Control	687
0x084	PWM1INTEN	R/W	0x0000.0000	PWM1 Interrupt and Trigger Enable	692
0x088	PWM1RIS	RO	0x0000.0000	PWM1 Raw Interrupt Status	694
0x08C	PWM1ISC	R/W1C	0x0000.0000	PWM1 Interrupt Status and Clear	695
0x090	PWM1LOAD	R/W	0x0000.0000	PWM1 Load	696
0x094	PWM1COUNT	RO	0x0000.0000	PWM1 Counter	697
0x098	PWM1CMPA	R/W	0x0000.0000	PWM1 Compare A	698
0x09C	PWM1CMPB	R/W	0x0000.0000	PWM1 Compare B	699
0x0A0	PWM1GENA	R/W	0x0000.0000	PWM1 Generator A Control	700
0x0A4	PWM1GENB	R/W	0x0000.0000	PWM1 Generator B Control	703
0x0A8	PWM1DBCTL	R/W	0x0000.0000	PWM1 Dead-Band Control	706
0x0AC	PWM1DBRISE	R/W	0x0000.0000	PWM1 Dead-Band Rising-Edge Delay	707
0x0B0	PWM1DBFALL	R/W	0x0000.0000	PWM1 Dead-Band Falling-Edge-Delay	708
0x0B4	PWM1FLTSRC0	R/W	0x0000.0000	PWM1 Fault Source 0	709
0x0BC	PWM1MINFLTPER	R/W	0x0000.0000	PWM1 Minimum Fault Period	711
0x0C0	PWM2CTL	R/W	0x0000.0000	PWM2 Control	687
0x0C4	PWM2INTEN	R/W	0x0000.0000	PWM2 Interrupt and Trigger Enable	692
0x0C8	PWM2RIS	RO	0x0000.0000	PWM2 Raw Interrupt Status	694
0x0CC	PWM2ISC	R/W1C	0x0000.0000	PWM2 Interrupt Status and Clear	695
0x0D0	PWM2LOAD	R/W	0x0000.0000	PWM2 Load	696
0x0D4	PWM2COUNT	RO	0x0000.0000	PWM2 Counter	697
0x0D8	PWM2CMPA	R/W	0x0000.0000	PWM2 Compare A	698
0x0DC	PWM2CMPB	R/W	0x0000.0000	PWM2 Compare B	699
0x0E0	PWM2GENA	R/W	0x0000.0000	PWM2 Generator A Control	700
0x0E4	PWM2GENB	R/W	0x0000.0000	PWM2 Generator B Control	703
0x0E8	PWM2DBCTL	R/W	0x0000.0000	PWM2 Dead-Band Control	706
0x0EC	PWM2DBRISE	R/W	0x0000.0000	PWM2 Dead-Band Rising-Edge Delay	707
0x0F0	PWM2DBFALL	R/W	0x0000.0000	PWM2 Dead-Band Falling-Edge-Delay	708
0x0F4	PWM2FLTSRC0	R/W	0x0000.0000	PWM2 Fault Source 0	709
0x0FC	PWM2MINFLTPER	R/W	0x0000.0000	PWM2 Minimum Fault Period	711
0x100	PWM3CTL	R/W	0x0000.0000	PWM3 Control	687
0x104	PWM3INTEN	R/W	0x0000.0000	PWM3 Interrupt and Trigger Enable	692
0x108	PWM3RIS	RO	0x0000.0000	PWM3 Raw Interrupt Status	694

Offset	Name	Type	Reset	Description	See page
0x10C	PWM3ISC	R/W1C	0x0000.0000	PWM3 Interrupt Status and Clear	695
0x110	PWM3LOAD	R/W	0x0000.0000	PWM3 Load	696
0x114	PWM3COUNT	RO	0x0000.0000	PWM3 Counter	697
0x118	PWM3CMPA	R/W	0x0000.0000	PWM3 Compare A	698
0x11C	PWM3CMPB	R/W	0x0000.0000	PWM3 Compare B	699
0x120	PWM3GENA	R/W	0x0000.0000	PWM3 Generator A Control	700
0x124	PWM3GENB	R/W	0x0000.0000	PWM3 Generator B Control	703
0x128	PWM3DBCTL	R/W	0x0000.0000	PWM3 Dead-Band Control	706
0x12C	PWM3DBRISE	R/W	0x0000.0000	PWM3 Dead-Band Rising-Edge Delay	707
0x130	PWM3DBFALL	R/W	0x0000.0000	PWM3 Dead-Band Falling-Edge-Delay	708
0x134	PWM3FLTSRC0	R/W	0x0000.0000	PWM3 Fault Source 0	709
0x13C	PWM3MINFLTPER	R/W	0x0000.0000	PWM3 Minimum Fault Period	711
0x800	PWM0FLTSEN	R/W	0x0000.0000	PWM0 Fault Pin Logic Sense	712
0x804	PWM0FLTSTAT0	-	0x0000.0000	PWM0 Fault Status 0	713
0x880	PWM1FLTSEN	R/W	0x0000.0000	PWM1 Fault Pin Logic Sense	712
0x884	PWM1FLTSTAT0	-	0x0000.0000	PWM1 Fault Status 0	713
0x900	PWM2FLTSEN	R/W	0x0000.0000	PWM2 Fault Pin Logic Sense	712
0x904	PWM2FLTSTAT0	-	0x0000.0000	PWM2 Fault Status 0	713
0x980	PWM3FLTSEN	R/W	0x0000.0000	PWM3 Fault Pin Logic Sense	712
0x984	PWM3FLTSTAT0	-	0x0000.0000	PWM3 Fault Status 0	713

20.5 Register Descriptions

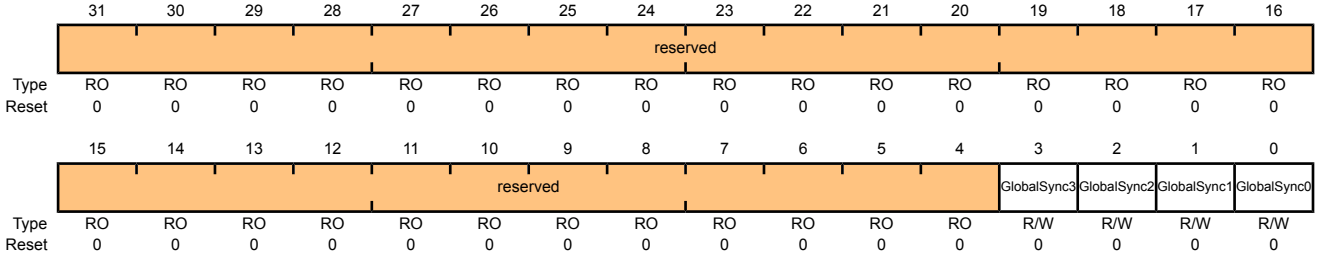
The remainder of this section lists and describes the PWM registers, in numerical order by address offset.

Register 1: PWM Master Control (PWMCTL), offset 0x000

This register provides master control over the PWM generation blocks.

PWM Master Control (PWMCTL)

Base 0x4002.8000
Offset 0x000
Type R/W, reset 0x0000.0000



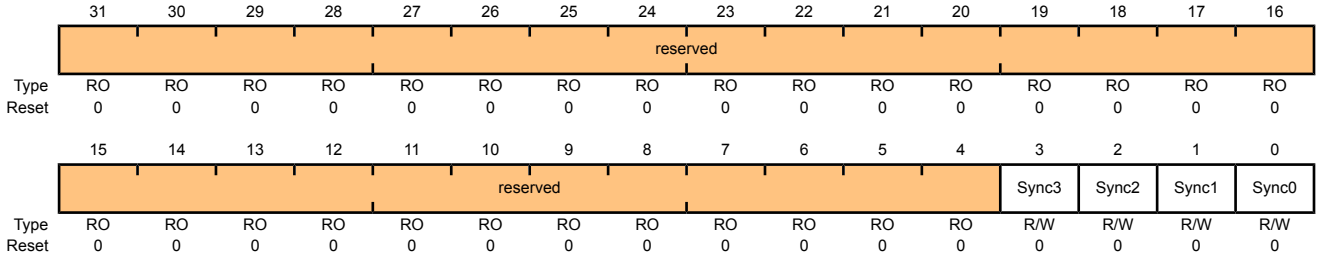
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	GlobalSync3	R/W	0	Update PWM Generator 3 Same as `GlobalSync0` but for PWM generator 3.
2	GlobalSync2	R/W	0	Update PWM Generator 2 Same as `GlobalSync0` but for PWM generator 2.
1	GlobalSync1	R/W	0	Update PWM Generator 1 Same as `GlobalSync0` but for PWM generator 1.
0	GlobalSync0	R/W	0	Update PWM Generator 0 Setting this bit causes any queued update to a load or comparator register in PWM generator 0 to be applied the next time the corresponding counter becomes zero. This bit automatically clears when the updates have completed; it cannot be cleared by software.

Register 2: PWM Time Base Sync (PWMSYNC), offset 0x004

This register provides a method to perform synchronization of the counters in the PWM generation blocks. Writing a bit in this register to 1 causes the specified counter to reset back to 0; writing multiple bits resets multiple counters simultaneously. The bits auto-clear after the reset has occurred; reading them back as zero indicates that the synchronization has completed.

PWM Time Base Sync (PWMSYNC)

Base 0x4002.8000
Offset 0x004
Type R/W, reset 0x0000.0000



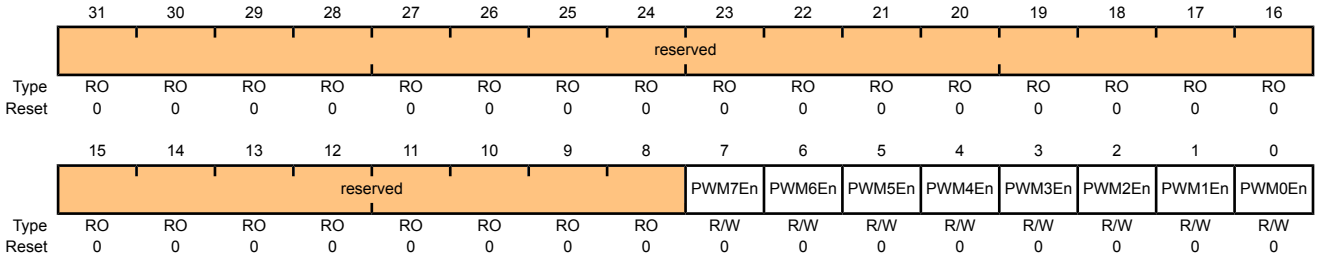
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	Sync3	R/W	0	Reset Generator 3 Counter Performs a reset of the PWM generator 3 counter.
2	Sync2	R/W	0	Reset Generator 2 Counter Performs a reset of the PWM generator 2 counter.
1	Sync1	R/W	0	Reset Generator 1 Counter Performs a reset of the PWM generator 1 counter.
0	Sync0	R/W	0	Reset Generator 0 Counter Performs a reset of the PWM generator 0 counter.

Register 3: PWM Output Enable (PWMENABLE), offset 0x008

This register provides a master control of which generated PWM signals are output to device pins. By disabling a PWM output, the generation process can continue (for example, when the time bases are synchronized) without driving PWM signals to the pins. When bits in this register are set, the corresponding PWM signal is passed through to the output stage, which is controlled by the **PWMINVERT** register. When bits are not set, the PWM signal is replaced by a zero value which is also passed to the output stage.

PWM Output Enable (PWMENABLE)

Base 0x4002.8000
Offset 0x008
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	PWM7En	R/W	0	PWM7 Output Enable When set, allows the generated PWM7 signal to be passed to the device pin.
6	PWM6En	R/W	0	PWM6 Output Enable When set, allows the generated PWM6 signal to be passed to the device pin.
5	PWM5En	R/W	0	PWM5 Output Enable When set, allows the generated PWM5 signal to be passed to the device pin.
4	PWM4En	R/W	0	PWM4 Output Enable When set, allows the generated PWM4 signal to be passed to the device pin.
3	PWM3En	R/W	0	PWM3 Output Enable When set, allows the generated PWM3 signal to be passed to the device pin.
2	PWM2En	R/W	0	PWM2 Output Enable When set, allows the generated PWM2 signal to be passed to the device pin.

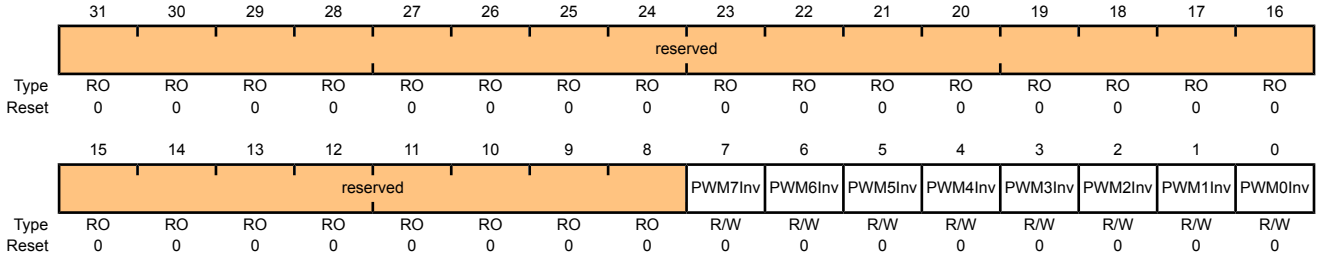
Bit/Field	Name	Type	Reset	Description
1	PWM1En	R/W	0	PWM1 Output Enable When set, allows the generated PWM1 signal to be passed to the device pin.
0	PWM0En	R/W	0	PWM0 Output Enable When set, allows the generated PWM0 signal to be passed to the device pin.

Register 4: PWM Output Inversion (PWMINVERT), offset 0x00C

This register provides a master control of the polarity of the PWM signals on the device pins. The PWM signals generated by the PWM generator are active High; they can optionally be made active Low via this register. Disabled PWM channels are also passed through the output inverter (if so configured) so that inactive channels maintain the correct polarity.

PWM Output Inversion (PWMINVERT)

Base 0x4002.8000
Offset 0x00C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	PWM7Inv	R/W	0	Invert PWM7 Signal When set, the generated PWM7 signal is inverted.
6	PWM6Inv	R/W	0	Invert PWM6 Signal When set, the generated PWM6 signal is inverted.
5	PWM5Inv	R/W	0	Invert PWM5 Signal When set, the generated PWM5 signal is inverted.
4	PWM4Inv	R/W	0	Invert PWM4 Signal When set, the generated PWM4 signal is inverted.
3	PWM3Inv	R/W	0	Invert PWM3 Signal When set, the generated PWM3 signal is inverted.
2	PWM2Inv	R/W	0	Invert PWM2 Signal When set, the generated PWM2 signal is inverted.
1	PWM1Inv	R/W	0	Invert PWM1 Signal When set, the generated PWM1 signal is inverted.
0	PWM0Inv	R/W	0	Invert PWM0 Signal When set, the generated PWM0 signal is inverted.

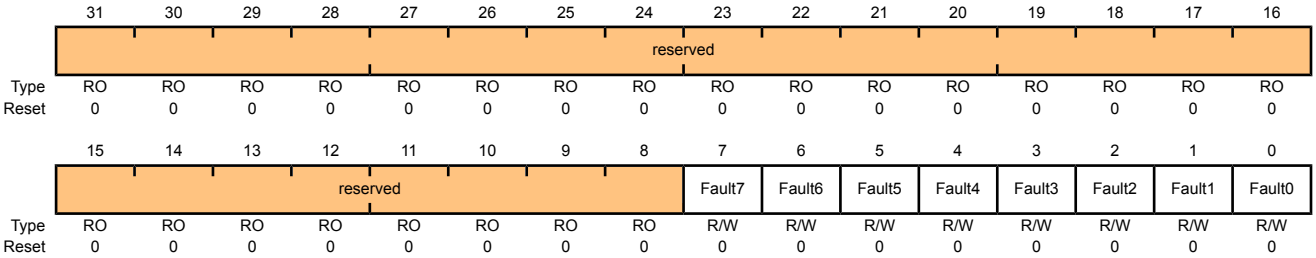
Register 5: PWM Output Fault (PWMFAULT), offset 0x010

This register controls the behavior of the PWM outputs in the presence of fault conditions. Both the fault inputs and debug events are considered fault conditions. On a fault condition, each PWM signal can be passed through unmodified or driven to a specified value. For outputs that are configured for pass-through, the debug event handling on the corresponding PWM generator also determines if the PWM signal continues to be generated.

Fault condition control occurs before the output inverter, so PWM signals driven to a specified value on fault are inverted if the channel is configured for inversion (therefore, the pin is driven to the logical complement of the specified value on a fault condition).

PWM Output Fault (PWMFAULT)

Base 0x4002.8000
Offset 0x010
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	Fault7	R/W	0	PWM7 Fault When set, the PWM7 output signal is driven to a specified value on a fault condition.
6	Fault6	R/W	0	PWM6 Fault When set, the PWM6 output signal is driven to a specified value on a fault condition.
5	Fault5	R/W	0	PWM5 Fault When set, the PWM5 output signal is driven to a specified value on a fault condition.
4	Fault4	R/W	0	PWM4 Fault When set, the PWM4 output signal is driven to a specified value on a fault condition.
3	Fault3	R/W	0	PWM3 Fault When set, the PWM3 output signal is driven to a specified value on a fault condition.
2	Fault2	R/W	0	PWM2 Fault When set, the PWM2 output signal is driven to a specified value on a fault condition.

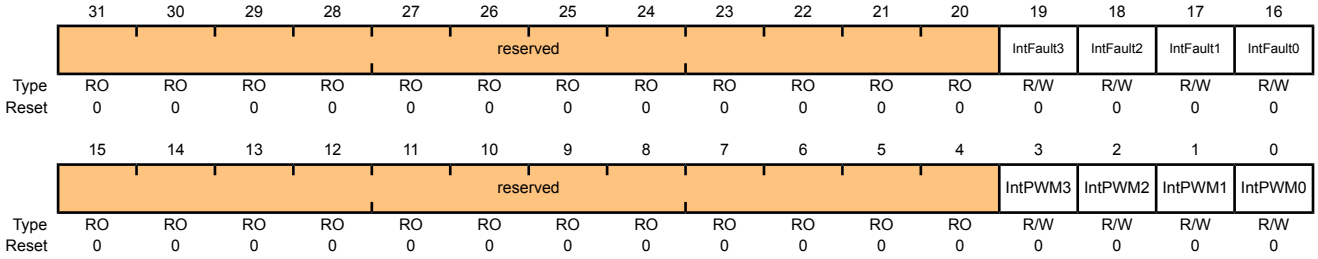
Bit/Field	Name	Type	Reset	Description
1	Fault1	R/W	0	PWM1 Fault When set, the PWM1 output signal is driven to a specified value on a fault condition.
0	Fault0	R/W	0	PWM0 Fault When set, the PWM0 output signal is driven to a specified value on a fault condition.

Register 6: PWM Interrupt Enable (PWMINTEN), offset 0x014

This register controls the global interrupt generation capabilities of the PWM module. The events that can cause an interrupt are the fault input and the individual interrupts from the PWM generators.

PWM Interrupt Enable (PWMINTEN)

Base 0x4002.8000
Offset 0x014
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:20	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	IntFault3	R/W	0	Interrupt Fault 3 When set, an interrupt occurs when the fault condition for PWM generator 3 is asserted.
18	IntFault2	R/W	0	Interrupt Fault 2 When set, an interrupt occurs when the fault condition for PWM generator 2 is asserted.
17	IntFault1	R/W	0	Interrupt Fault 1 When set, an interrupt occurs when the fault condition for PWM generator 1 is asserted.
16	IntFault0	R/W	0	Interrupt Fault 0 When set, an interrupt occurs when the FAULT0 input is asserted or the fault condition for PWM generator 0 is asserted.
15:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IntPWM3	R/W	0	PWM3 Interrupt Enable When set, an interrupt occurs when the PWM generator 3 block asserts an interrupt.
2	IntPWM2	R/W	0	PWM2 Interrupt Enable When set, an interrupt occurs when the PWM generator 2 block asserts an interrupt.
1	IntPWM1	R/W	0	PWM1 Interrupt Enable When set, an interrupt occurs when the PWM generator 1 block asserts an interrupt.

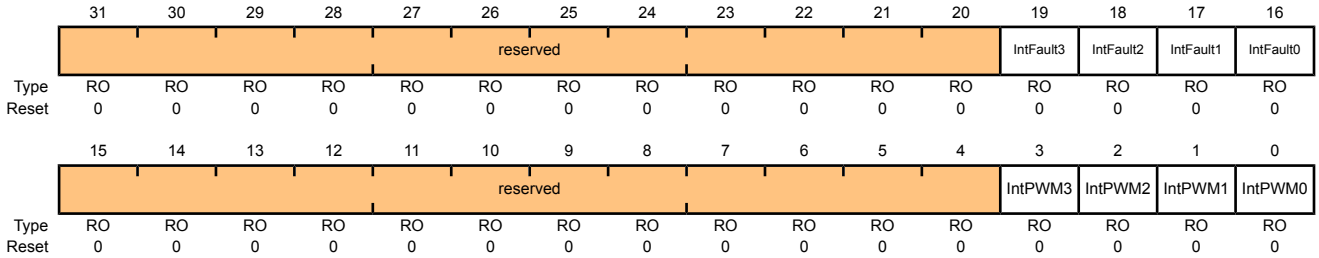
Bit/Field	Name	Type	Reset	Description
0	IntPWM0	R/W	0	PWM0 Interrupt Enable When set, an interrupt occurs when the PWM generator 0 block asserts an interrupt.

Register 7: PWM Raw Interrupt Status (PWMRIS), offset 0x018

This register provides the current set of interrupt sources that are asserted, regardless of whether they cause an interrupt to be asserted to the controller. The fault interrupt is latched on detection; it must be cleared through the **PWM Interrupt Status and Clear (PWMISC)** register (see page 682). The PWM generator interrupts simply reflect the status of the PWM generators; they are cleared via the interrupt status register in the PWM generator blocks. Bits set to 1 indicate the events that are active; zero bits indicate that the event in question is not active.

PWM Raw Interrupt Status (PWMRIS)

Base 0x4002.8000
Offset 0x018
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:20	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	IntFault3	RO	0	Interrupt Fault PWM 3 Indicates that the fault condition for PWM generator 3 is asserting.
18	IntFault2	RO	0	Interrupt Fault PWM 2 Indicates that the fault condition for PWM generator 2 is asserting.
17	IntFault1	RO	0	Interrupt Fault PWM 1 Indicates that the fault condition for PWM generator 1 is asserting.
16	IntFault0	RO	0	Interrupt Fault PWM 0 Indicates that the FAULT0 input is asserting or the fault condition for PWM generator 0 is asserting.
15:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IntPWM3	RO	0	PWM3 Interrupt Asserted Indicates that the PWM generator 3 block is asserting its interrupt.
2	IntPWM2	RO	0	PWM2 Interrupt Asserted Indicates that the PWM generator 2 block is asserting its interrupt.
1	IntPWM1	RO	0	PWM1 Interrupt Asserted Indicates that the PWM generator 1 block is asserting its interrupt.

Bit/Field	Name	Type	Reset	Description
0	IntPWM0	RO	0	PWM0 Interrupt Asserted Indicates that the PWM generator 0 block is asserting its interrupt.

Register 8: PWM Interrupt Status and Clear (PWMISC), offset 0x01C

This register provides a summary of the interrupt status of the individual PWM generator blocks. A bit set to 1 indicates that the corresponding generator block is asserting an interrupt. The individual interrupt status registers in each block must be consulted to determine the reason for the interrupt, and used to clear the interrupt. For the fault interrupt, a write of 1 to that bit position clears the latched interrupt status.

PWM Interrupt Status and Clear (PWMISC)

Base 0x4002.8000
Offset 0x01C
Type R/W1C, reset 0x0000.0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
reserved												IntFault3	IntFault2	IntFault1	IntFault0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W1C	R/W1C	R/W1C	R/W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved												IntPWM3	IntPWM2	IntPWM1	IntPWM0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:20	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
19	IntFault3	R/W1C	0	FAULT3 Interrupt Asserted Indicates that the FAULT3 input is asserting or the FAULT3 latch has captured an assertion.
18	IntFault2	R/W1C	0	FAULT2 Interrupt Asserted Indicates that the FAULT2 input is asserting or the FAULT2 latch has captured an assertion.
17	IntFault1	R/W1C	0	FAULT1 Interrupt Asserted Indicates that the FAULT1 input is asserting or the FAULT1 latch has captured an assertion.
16	IntFault0	R/W1C	0	FAULT0 Interrupt Asserted Indicates that the FAULT0 input is asserting or the fault condition for generator 0 is assertng a fault.
15:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IntPWM3	RO	0	PWM3 Interrupt Status Indicates if the PWM generator 3 block is asserting an interrupt.
2	IntPWM2	RO	0	PWM2 Interrupt Status Indicates if the PWM generator 2 block is asserting an interrupt.
1	IntPWM1	RO	0	PWM1 Interrupt Status Indicates if the PWM generator 1 block is asserting an interrupt.

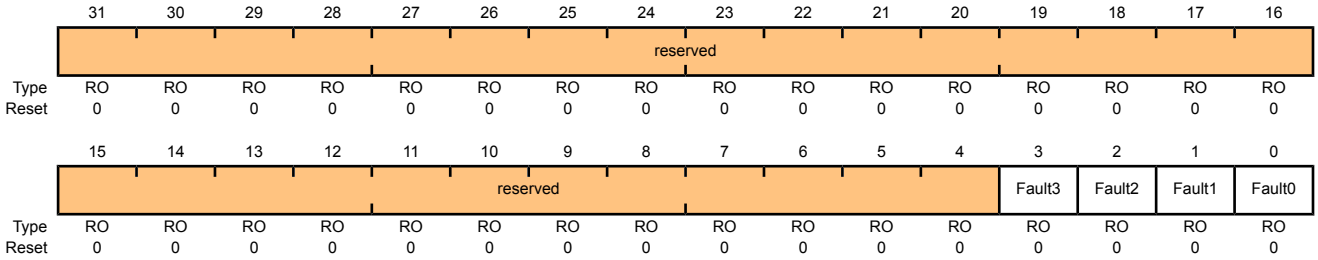
Bit/Field	Name	Type	Reset	Description
0	IntPWM0	RO	0	PWM0 Interrupt Status Indicates if the PWM generator 0 block is asserting an interrupt.

Register 9: PWM Status (PWMSTATUS), offset 0x020

This register provides the status of the FAULT input signals.

PWM Status (PWMSTATUS)

Base 0x4002.8000
Offset 0x020
Type RO, reset 0x0000.0000



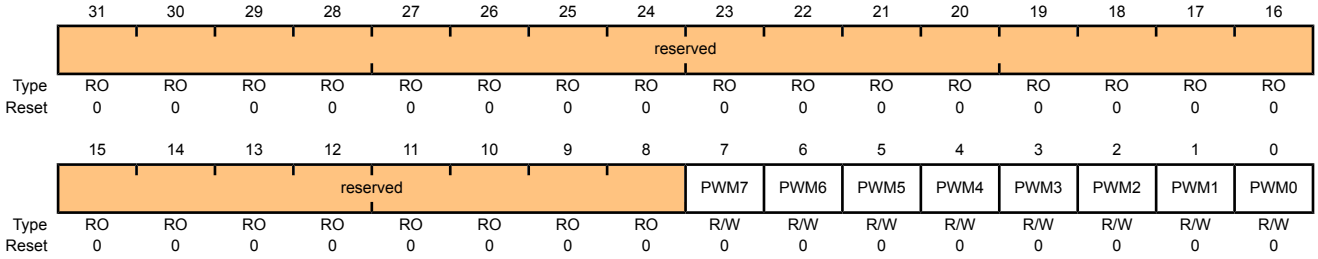
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	Fault3	RO	0	Fault3 Interrupt Status When set, indicates the fault condition for PWM generator 3 is asserted.
2	Fault2	RO	0	Fault2 Interrupt Status When set, indicates the fault condition for PWM generator 2 is asserted.
1	Fault1	RO	0	Fault1 Interrupt Status When set, indicates the fault condition for PWM generator 1 is asserted.
0	Fault0	RO	0	Fault0 Interrupt Status When set, indicates the FAULT0 input is asserted, or that the fault condition for PWM generator 0 is asserted.

Register 10: PWM Fault Condition Value (PWMFAULTVAL), offset 0x024

This register specifies the output value driven on the PWM signals during a fault condition if the corresponding bit in the **PWMFAULT** register is indicating that the PWM signal drives a value.

PWM Fault Condition Value (PWMFAULTVAL)

Base 0x4002.8000
Offset 0x024
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
7	PWM7	R/W	0	PWM7 Fault Value The PWM7 output signal is driven to the value specified in this bit during fault conditions if the Fault7 bit in the **PWMFAULT** register is set.
6	PWM6	R/W	0	PWM6 Fault Value The PWM6 output signal is driven to the value specified in this bit during fault conditions if the Fault6 bit in the **PWMFAULT** register is set.
5	PWM5	R/W	0	PWM5 Fault Value The PWM5 output signal is driven to the value specified in this bit during fault conditions if the Fault5 bit in the **PWMFAULT** register is set.
4	PWM4	R/W	0	PWM4 Fault Value The PWM4 output signal is driven to the value specified in this bit during fault conditions if the Fault4 bit in the **PWMFAULT** register is set.
3	PWM3	R/W	0	PWM3 Fault Value The PWM3 output signal is driven to the value specified in this bit during fault conditions if the Fault3 bit in the **PWMFAULT** register is set.
2	PWM2	R/W	0	PWM2 Fault Value The PWM2 output signal is driven to the value specified in this bit during fault conditions if the Fault2 bit in the **PWMFAULT** register is set.
1	PWM1	R/W	0	PWM1 Fault Value The PWM1 output signal is driven to the value specified in this bit during fault conditions if the Fault1 bit in the **PWMFAULT** register is set.

Bit/Field	Name	Type	Reset	Description
0	PWM0	R/W	0	PWM0 Fault Value The PWM0 output signal is driven to the value specified in this bit during fault conditions if the Fault0 bit in the **PWMFAULT** register is set.

Register 11: PWM0 Control (PWM0CTL), offset 0x040

Register 12: PWM1 Control (PWM1CTL), offset 0x080

Register 13: PWM2 Control (PWM2CTL), offset 0x0C0

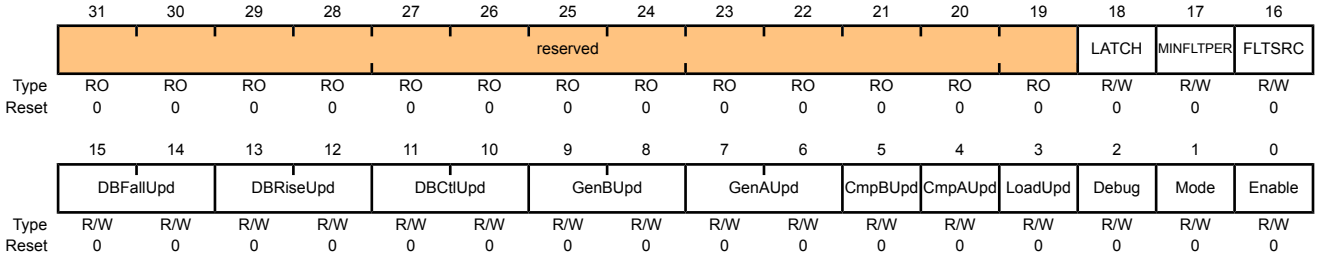
Register 14: PWM3 Control (PWM3CTL), offset 0x100

These registers configure the PWM signal generation blocks (PWM0CTL controls the PWM generator 0 block, and so on). The Register Update mode, Debug mode, Counting mode, and Block Enable mode are all controlled via these registers. The blocks produce the PWM signals, which can be either two independent PWM signals (from the same counter), or a paired set of PWM signals with dead-band delays added.

The PWM0 block produces the PWM0 and PWM1 outputs, the PWM1 block produces the PWM2 and PWM3 outputs, the PWM2 block produces the PWM4 and PWM5 outputs, and the PWM3 block produces the PWM6 and PWM7 outputs.

PWM0 Control (PWM0CTL)

Base 0x4002.8000
Offset 0x040
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:19	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
18	LATCH	R/W	0	Latch Fault Input This bit controls the behavior of the fault condition in a PWM generator. The fault condition may be latched and internally asserted because the fault condition logic includes the generator's IntFaultn bit (of the **PWMISC** register) enabled by the LATCH bit. Therefore, if the **PWMINTEN** IntFaultn bit is set, a fault condition sets the **PWMISC** IntFaultn bit (generating an interrupt) and the fault condition is extended in the generator logic until software clears the **PWMISC** IntFaultn bit. Value Description 0 Fault Condition Not Latched A fault condition is in effect for as long as the generating source is asserting. 1 Fault Condition Latched A fault condition is set as the result of the assertion of the faulting source and is held (latched) while the **PWMISC** IntFaultn bit is set. Clearing the IntFaultn bit clears the fault condition.
17	MINFLTPER	R/W	0	Minimum Fault Period This bit specifies that the PWM generator enables a one-shot counter to provide a minimum fault condition period. The timer begins counting on the rising edge of the fault condition to extend the condition for a minimum duration of the count value. The timer ignores the state of the fault condition while counting. The minimum fault delay is in effect only when the MINFLTPER bit is set. If a detected fault is in the process of being extended when the MINFLTPER bit is cleared, the fault condition extension is aborted. The delay time is specified by the **PWMnMINFLTPER** register MFP field value. The effect of this is to pulse stretch the fault condition input. The delay value is defined by the PWM clock period. Because the fault input is not synchronized to the PWM clock, the period of the time is PWMClock * (MFP value + 1) or PWMClock * (MFP value + 2). The delay function makes sense only if the fault source is unlatched. A latched fault source makes the fault condition appear asserted until cleared by software and negates the utility of the extend feature. It applies to all fault condition sources as specified in the FLTSRC field. Value Description 0 Fault Condition Period Not Extended The FAULT input deassertion is unaffected. 1 Fault Condition Period Extended The **PWMnMINFLTPER** one-shot counter is active and extends the period of the fault condition to a minimum period.

Bit/Field	Name	Type	Reset	Description
16	FLTSRC	R/W	0	Fault Condition Source This bit specifies the fault condition source. Value Description 0 Fault0 The Fault condition is determined by the Fault0 input. 1 Register-Defined The Fault condition is determined by the configuration of the **PWMnFLTSRC0** register.
15:14	DBFallUpd	R/W	0	**PWMnDBFALL** Update Mode Specifies the update mode for the **PWMnDBFALL** register. Value Description 0 Immediate The **PWMnDBFALL** register value is immediately updated on a write. 1 Reserved 2 Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0. 3 Globally Synchronized Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (**PWMCTL**) register.
13:12	DBRiseUpd	R/W	0	**PWMnDBRISE** Update Mode Specifies the update mode for the **PWMnDBRISE** register. Value Description 0 Immediate The **PWMnDBRISE** register value is immediately updated on a write. 1 Reserved 2 Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0. 3 Globally Synchronized Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (**PWMCTL**) register.

Bit/Field	Name	Type	Reset	Description
11:10	DBCtlUpd	R/W	0	**PWMnDBCTL** Update Mode Specifies the update mode for the **PWMnDBCTL** register. Value Description 0 Immediate The **PWMnDBCTL** register value is immediately updated on a write. 1 Reserved 2 Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0. 3 Globally Synchronized Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (**PWMCTL**) register.
9:8	GenBUpd	R/W	0	**PWMnGENB** Update Mode Specifies the update mode for the **PWMnGENB** register. Value Description 0 Immediate The **PWMnGENB** register value is immediately updated on a write. 1 Reserved 2 Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0. 3 Globally Synchronized Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (**PWMCTL**) register.

Bit/Field	Name	Type	Reset	Description
7:6	GenAUpd	R/W	0	**PWMnGENA** Update Mode Specifies the update mode for the **PWMnGENA** register. Value Description 0 Immediate The **PWMnGENA** register value is immediately updated on a write. 1 Reserved 2 Locally Synchronized Updates to the register are reflected to the generator the next time the counter is 0. 3 Globally Synchronized Updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the PWM Master Control (**PWMCTL**) register.
5	CmpBUpd	R/W	0	Comparator B Update Mode Same as `CmpAUpd` but for the comparator B register.
4	CmpAUpd	R/W	0	Comparator A Update Mode The Update mode for the comparator A register. When not set, updates to the register are reflected to the comparator the next time the counter is 0. When set, updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 671).
3	LoadUpd	R/W	0	Load Register Update Mode The Update mode for the load register. When not set, updates to the register are reflected to the counter the next time the counter is 0. When set, updates to the register are delayed until the next time the counter is 0 after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register.
2	Debug	R/W	0	Debug Mode The behavior of the counter in Debug mode. When not set, the counter stops running when it next reaches 0, and continues running again when no longer in Debug mode. When set, the counter always runs.
1	Mode	R/W	0	Counter Mode The mode for the counter. When not set, the counter counts down from the load value to 0 and then wraps back to the load value (Count-Down mode). When set, the counter counts up from 0 to the load value, back down to 0, and then repeats (Count-Up/Down mode).
0	Enable	R/W	0	PWM Block Enable Master enable for the PWM generation block. When not set, the entire block is disabled and not clocked. When set, the block is enabled and produces PWM signals.

Register 15: PWM0 Interrupt and Trigger Enable (PWM0INTEN), offset 0x044
Register 16: PWM1 Interrupt and Trigger Enable (PWM1INTEN), offset 0x084
Register 17: PWM2 Interrupt and Trigger Enable (PWM2INTEN), offset 0x0C4
Register 18: PWM3 Interrupt and Trigger Enable (PWM3INTEN), offset 0x104

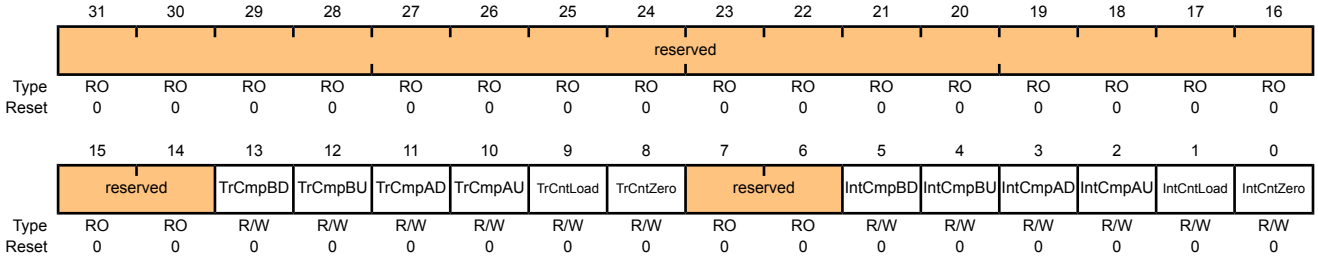
These registers control the interrupt and ADC trigger generation capabilities of the PWM generators (**PWM0INTEN** controls the PWM generator 0 block, and so on). The events that can cause an interrupt or an ADC trigger are:

- The counter being equal to the load register
- The counter being equal to zero
- The counter being equal to the comparator A register while counting up
- The counter being equal to the comparator A register while counting down
- The counter being equal to the comparator B register while counting up
- The counter being equal to the comparator B register while counting down

Any combination of these events can generate either an interrupt, or an ADC trigger; though no determination can be made as to the actual event that caused an ADC trigger if more than one is specified.

PWM0 Interrupt and Trigger Enable (PWM0INTEN)

Base 0x4002.8000
Offset 0x044
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:14	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
13	TrCmpBD	R/W	0	Trigger for Counter=Comparator B Down When 1, a trigger pulse is output when the counter matches the comparator B value and the counter is counting down.
12	TrCmpBU	R/W	0	Trigger for Counter=Comparator B Up When 1, a trigger pulse is output when the counter matches the comparator B value and the counter is counting up.

Bit/Field	Name	Type	Reset	Description
11	TrCmpAD	R/W	0	Trigger for Counter=Comparator A Down When 1, a trigger pulse is output when the counter matches the comparator A value and the counter is counting down.
10	TrCmpAU	R/W	0	Trigger for Counter=Comparator A Up When 1, a trigger pulse is output when the counter matches the comparator A value and the counter is counting up.
9	TrCntLoad	R/W	0	Trigger for Counter=Load When 1, a trigger pulse is output when the counter matches the **PWMnLOAD** register.
8	TrCntZero	R/W	0	Trigger for Counter=0 When 1, a trigger pulse is output when the counter is 0.
7:6	reserved	RO	0x0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	IntCmpBD	R/W	0	Interrupt for Counter=Comparator B Down When 1, an interrupt occurs when the counter matches the comparator B value and the counter is counting down.
4	IntCmpBU	R/W	0	Interrupt for Counter=Comparator B Up When 1, an interrupt occurs when the counter matches the comparator B value and the counter is counting up.
3	IntCmpAD	R/W	0	Interrupt for Counter=Comparator A Down When 1, an interrupt occurs when the counter matches the comparator A value and the counter is counting down.
2	IntCmpAU	R/W	0	Interrupt for Counter=Comparator A Up When 1, an interrupt occurs when the counter matches the comparator A value and the counter is counting up.
1	IntCntLoad	R/W	0	Interrupt for Counter=Load When 1, an interrupt occurs when the counter matches the **PWMnLOAD** register.
0	IntCntZero	R/W	0	Interrupt for Counter=0 When 1, an interrupt occurs when the counter is 0.

Register 19: PWM0 Raw Interrupt Status (PWM0RIS), offset 0x048

Register 20: PWM1 Raw Interrupt Status (PWM1RIS), offset 0x088

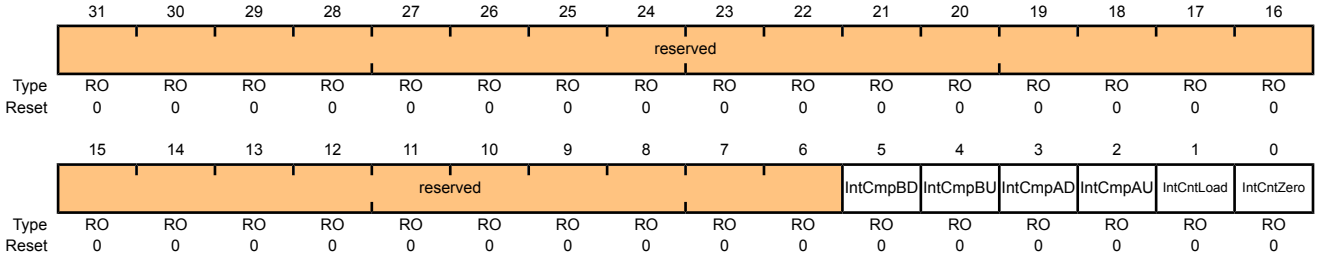
Register 21: PWM2 Raw Interrupt Status (PWM2RIS), offset 0x0C8

Register 22: PWM3 Raw Interrupt Status (PWM3RIS), offset 0x108

These registers provide the current set of interrupt sources that are asserted, regardless of whether they cause an interrupt to be asserted to the controller (**PWM0RIS** controls the PWM generator 0 block, and so on). Bits set to 1 indicate the latched events that have occurred; bits set to 0 indicate that the event in question has not occurred.

PWM0 Raw Interrupt Status (PWM0RIS)

Base 0x4002.8000
Offset 0x048
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	IntCmpBD	RO	0	Comparator B Down Interrupt Status Indicates that the counter has matched the comparator B value while counting down.
4	IntCmpBU	RO	0	Comparator B Up Interrupt Status Indicates that the counter has matched the comparator B value while counting up.
3	IntCmpAD	RO	0	Comparator A Down Interrupt Status Indicates that the counter has matched the comparator A value while counting down.
2	IntCmpAU	RO	0	Comparator A Up Interrupt Status Indicates that the counter has matched the comparator A value while counting up.
1	IntCntLoad	RO	0	Counter=Load Interrupt Status Indicates that the counter has matched the **PWMnLOAD** register.
0	IntCntZero	RO	0	Counter=0 Interrupt Status Indicates that the counter has matched 0.

Register 23: PWM0 Interrupt Status and Clear (PWM0ISC), offset 0x04C

Register 24: PWM1 Interrupt Status and Clear (PWM1ISC), offset 0x08C

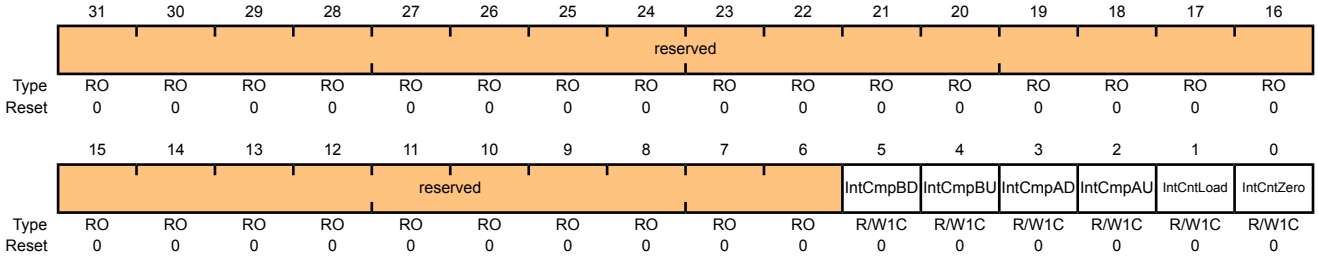
Register 25: PWM2 Interrupt Status and Clear (PWM2ISC), offset 0x0CC

Register 26: PWM3 Interrupt Status and Clear (PWM3ISC), offset 0x10C

These registers provide the current set of interrupt sources that are asserted to the controller (**PWM0ISC** controls the PWM generator 0 block, and so on). Bits set to 1 indicate the latched events that have occurred; bits set to 0 indicate that the event in question has not occurred. These are R/W1C registers; writing a 1 to a bit position clears the corresponding interrupt reason.

PWM0 Interrupt Status and Clear (PWM0ISC)

Base 0x4002.8000
Offset 0x04C
Type R/W1C, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
5	IntCmpBD	R/W1C	0	Comparator B Down Interrupt Indicates that the counter has matched the comparator B value while counting down.
4	IntCmpBU	R/W1C	0	Comparator B Up Interrupt Indicates that the counter has matched the comparator B value while counting up.
3	IntCmpAD	R/W1C	0	Comparator A Down Interrupt Indicates that the counter has matched the comparator A value while counting down.
2	IntCmpAU	R/W1C	0	Comparator A Up Interrupt Indicates that the counter has matched the comparator A value while counting up.
1	IntCntLoad	R/W1C	0	Counter=Load Interrupt Indicates that the counter has matched the **PWMnLOAD** register.
0	IntCntZero	R/W1C	0	Counter=0 Interrupt Indicates that the counter has matched 0.

Register 27: PWM0 Load (PWM0LOAD), offset 0x050

Register 28: PWM1 Load (PWM1LOAD), offset 0x090

Register 29: PWM2 Load (PWM2LOAD), offset 0x0D0

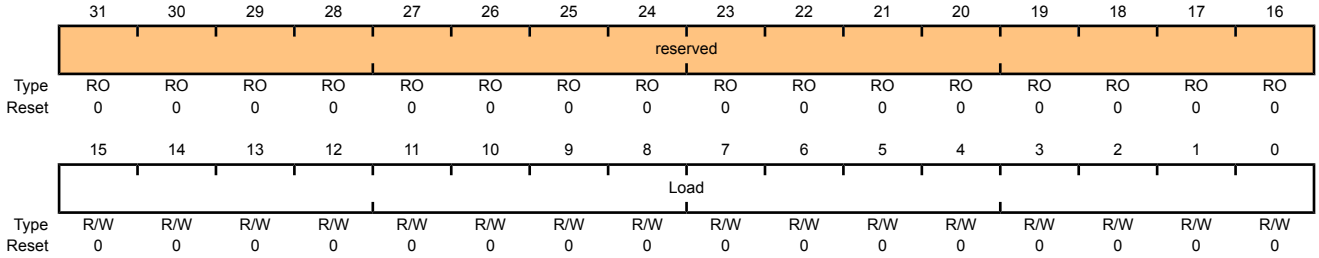
Register 30: PWM3 Load (PWM3LOAD), offset 0x110

These registers contain the load value for the PWM counter (**PWM0LOAD** controls the PWM generator 0 block, and so on). Based on the counter mode, either this value is loaded into the counter after it reaches zero, or it is the limit of up-counting after which the counter decrements back to zero.

If the Load Value Update mode is immediate, this value is used the next time the counter reaches zero; if the mode is synchronous, it is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 671). If this register is re-written before the actual update occurs, the previous value is never used and is lost.

PWM0 Load (PWM0LOAD)

Base 0x4002.8000
Offset 0x050
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	Load	R/W	0	Counter Load Value The counter load value.

Register 31: PWM0 Counter (PWM0COUNT), offset 0x054

Register 32: PWM1 Counter (PWM1COUNT), offset 0x094

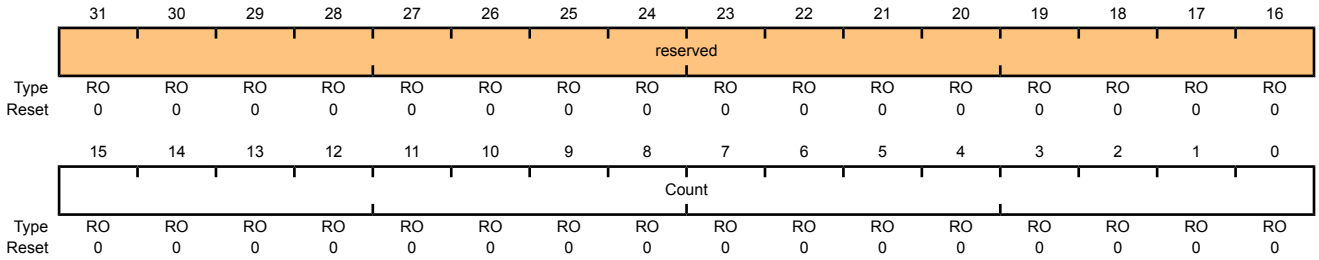
Register 33: PWM2 Counter (PWM2COUNT), offset 0x0D4

Register 34: PWM3 Counter (PWM3COUNT), offset 0x114

These registers contain the current value of the PWM counter. When this value matches the load register, a pulse is output; this can drive the generation of a PWM signal (via the **PWMnGENA/PWMnGENB** registers, see page 700 and page 703) or drive an interrupt or ADC trigger (via the **PWMnINTEN** register, see page 692). A pulse with the same capabilities is generated when this value is zero.

PWM0 Counter (PWM0COUNT)

Base 0x4002.8000
Offset 0x054
Type RO, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	Count	RO	0x00	Counter Value The current value of the counter.

Register 35: PWM0 Compare A (PWM0CMPA), offset 0x058

Register 36: PWM1 Compare A (PWM1CMPA), offset 0x098

Register 37: PWM2 Compare A (PWM2CMPA), offset 0x0D8

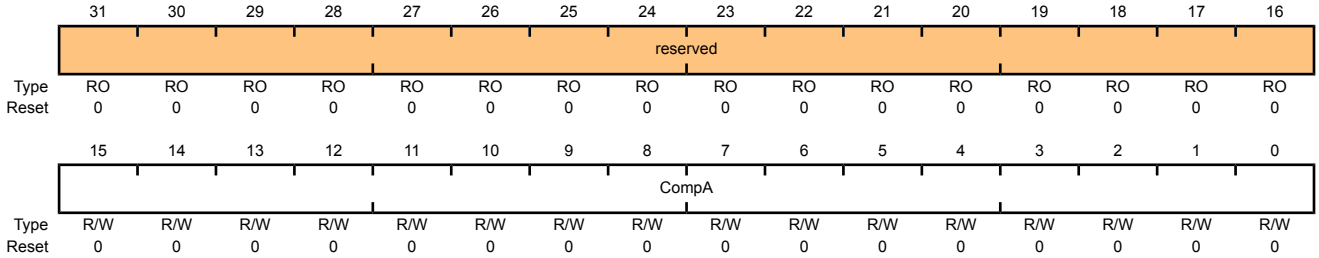
Register 38: PWM3 Compare A (PWM3CMPA), offset 0x118

These registers contain a value to be compared against the counter (**PWM0CMPA** controls the PWM generator 0 block, and so on). When this value matches the counter, a pulse is output; this can drive the generation of a PWM signal (via the **PWMnGENA/PWMnGENB** registers) or drive an interrupt or ADC trigger (via the **PWMnINTEN** register). If the value of this register is greater than the **PWMnLOAD** register (see page 696), then no pulse is ever output.

If the comparator A update mode is immediate (based on the CmpAUpd bit in the **PWMnCTL** register), this 16-bit CompA value is used the next time the counter reaches zero. If the update mode is synchronous, it is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 671). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWM0 Compare A (PWM0CMPA)

Base 0x4002.8000
Offset 0x058
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	CompA	R/W	0x00	Comparator A Value The value to be compared against the counter.

Register 39: PWM0 Compare B (PWM0CMPB), offset 0x05C

Register 40: PWM1 Compare B (PWM1CMPB), offset 0x09C

Register 41: PWM2 Compare B (PWM2CMPB), offset 0x0DC

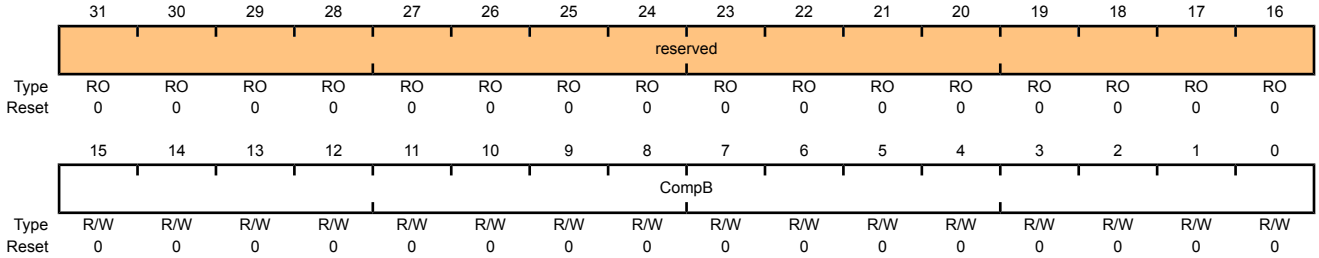
Register 42: PWM3 Compare B (PWM3CMPB), offset 0x11C

These registers contain a value to be compared against the counter (**PWM0CMPB** controls the PWM generator 0 block, and so on). When this value matches the counter, a pulse is output; this can drive the generation of a PWM signal (via the **PWMnGENA/PWMnGENB** registers) or drive an interrupt or ADC trigger (via the **PWMnINTEN** register). If the value of this register is greater than the **PWMnLOAD** register, no pulse is ever output.

If the comparator B update mode is immediate (based on the CmpBUpd bit in the **PWMnCTL** register), this 16-bit CompB value is used the next time the counter reaches zero. If the update mode is synchronous, it is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 671). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWM0 Compare B (PWM0CMPB)

Base 0x4002.8000
Offset 0x05C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	CompB	R/W	0x00	Comparator B Value The value to be compared against the counter.

Register 43: PWM0 Generator A Control (PWM0GENA), offset 0x060

Register 44: PWM1 Generator A Control (PWM1GENA), offset 0x0A0

Register 45: PWM2 Generator A Control (PWM2GENA), offset 0x0E0

Register 46: PWM3 Generator A Control (PWM3GENA), offset 0x120

These registers control the generation of the PWMnA signal based on the load and zero output pulses from the counter, as well as the compare A and compare B pulses from the comparators (**PWM0GENA** controls the PWM generator 0 block, and so on). When the counter is running in Count-Down mode, only four of these events occur; when running in Count-Up/Down mode, all six occur. These events provide great flexibility in the positioning and duty cycle of the PWM signal that is produced.

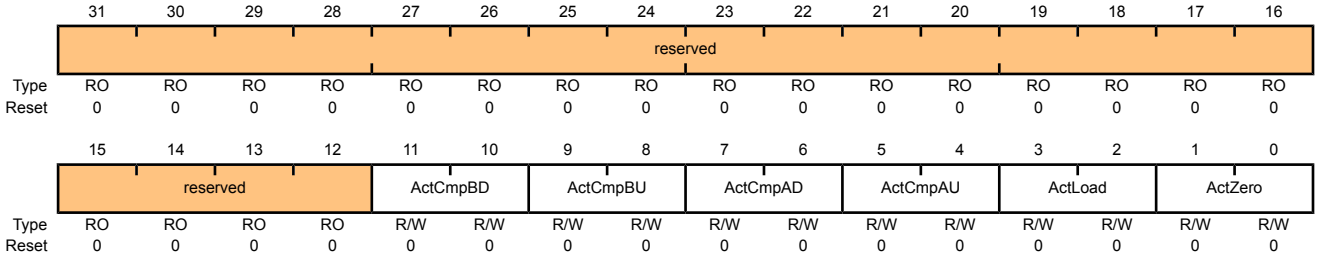
The **PWM0GENA** register controls generation of the PWM0A signal; **PWM1GENA**, the PWM1A signal; **PWM2GENA**, the PWM2A signal; and **PWM3GENA**, the PWM3A signal.

If a zero or load event coincides with a compare A or compare B event, the zero or load action is taken and the compare A or compare B action is ignored. If a compare A event coincides with a compare B event, the compare A action is taken and the compare B action is ignored.

If the Generator A update mode is immediate (based on the GenAUpd field encoding in the **PWMnCTL** register), this 16-bit GenAUpd value is used the next time the counter reaches zero. If the update mode is synchronous, it is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 671). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWM0 Generator A Control (PWM0GENA)

Base 0x4002.8000
Offset 0x060
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
11:10	ActCmpBD	R/W	0x0	Action for Comparator B Down The action to be taken when the counter matches comparator B while counting down. The table below defines the effect of the event on the output signal. Value Description 0x0 Do nothing. 0x1 Invert the output signal. 0x2 Set the output signal to 0. 0x3 Set the output signal to 1.
9:8	ActCmpBU	R/W	0x0	Action for Comparator B Up The action to be taken when the counter matches comparator B while counting up. Occurs only when the Mode bit in the **PWMnCTL** register (see page 687) is set to 1. The table below defines the effect of the event on the output signal. Value Description 0x0 Do nothing. 0x1 Invert the output signal. 0x2 Set the output signal to 0. 0x3 Set the output signal to 1.
7:6	ActCmpAD	R/W	0x0	Action for Comparator A Down The action to be taken when the counter matches comparator A while counting down. The table below defines the effect of the event on the output signal. Value Description 0x0 Do nothing. 0x1 Invert the output signal. 0x2 Set the output signal to 0. 0x3 Set the output signal to 1.
5:4	ActCmpAU	R/W	0x0	Action for Comparator A Up The action to be taken when the counter matches comparator A while counting up. Occurs only when the Mode bit in the **PWMnCTL** register is set to 1. The table below defines the effect of the event on the output signal. Value Description 0x0 Do nothing. 0x1 Invert the output signal. 0x2 Set the output signal to 0. 0x3 Set the output signal to 1.

Bit/Field	Name	Type	Reset	Description
3:2	ActLoad	R/W	0x0	Action for Counter=Load The action to be taken when the counter matches the load value. The table below defines the effect of the event on the output signal. Value Description 0x0 Do nothing. 0x1 Invert the output signal. 0x2 Set the output signal to 0. 0x3 Set the output signal to 1.
1:0	ActZero	R/W	0x0	Action for Counter=0 The action to be taken when the counter is zero. The table below defines the effect of the event on the output signal. Value Description 0x0 Do nothing. 0x1 Invert the output signal. 0x2 Set the output signal to 0. 0x3 Set the output signal to 1.

Register 47: PWM0 Generator B Control (PWM0GENB), offset 0x064

Register 48: PWM1 Generator B Control (PWM1GENB), offset 0x0A4

Register 49: PWM2 Generator B Control (PWM2GENB), offset 0x0E4

Register 50: PWM3 Generator B Control (PWM3GENB), offset 0x124

These registers control the generation of the PWMnB signal based on the load and zero output pulses from the counter, as well as the compare A and compare B pulses from the comparators (**PWM0GENB** controls the PWM generator 0 block, and so on). When the counter is running in Down mode, only four of these events occur; when running in Up/Down mode, all six occur. These events provide great flexibility in the positioning and duty cycle of the PWM signal that is produced.

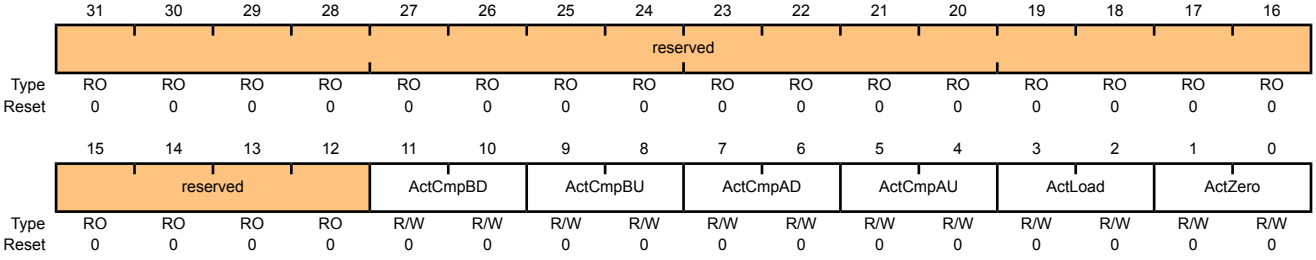
The **PWM0GENB** register controls generation of the PWM0B signal; **PWM1GENB**, the PWM1B signal; **PWM2GENB**, the PWM2B signal; and **PWM3GENB**, the PWM3B signal.

If a zero or load event coincides with a compare A or compare B event, the zero or load action is taken and the compare A or compare B action is ignored. If a compare A event coincides with a compare B event, the compare B action is taken and the compare A action is ignored.

If the Generator B update mode is immediate (based on the GenBUpd field encoding in the **PWMnCTL** register), this 16-bit GenBUpd value is used the next time the counter reaches zero. If the update mode is synchronous, it is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 671). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWM0 Generator B Control (PWM0GENB)

Base 0x4002.8000
Offset 0x064
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.

Bit/Field	Name	Type	Reset	Description
11:10	ActCmpBD	R/W	0x0	Action for Comparator B Down The action to be taken when the counter matches comparator B while counting down. The table below defines the effect of the event on the output signal. Value Description 0x0 Do nothing. 0x1 Invert the output signal. 0x2 Set the output signal to 0. 0x3 Set the output signal to 1.
9:8	ActCmpBU	R/W	0x0	Action for Comparator B Up The action to be taken when the counter matches comparator B while counting up. Occurs only when the `Mode` bit in the **PWMnCTL** register is set to 1. The table below defines the effect of the event on the output signal. Value Description 0x0 Do nothing. 0x1 Invert the output signal. 0x2 Set the output signal to 0. 0x3 Set the output signal to 1.
7:6	ActCmpAD	R/W	0x0	Action for Comparator A Down The action to be taken when the counter matches comparator A while counting down. The table below defines the effect of the event on the output signal. Value Description 0x0 Do nothing. 0x1 Invert the output signal. 0x2 Set the output signal to 0. 0x3 Set the output signal to 1.
5:4	ActCmpAU	R/W	0x0	Action for Comparator A Up The action to be taken when the counter matches comparator A while counting up. Occurs only when the `Mode` bit in the **PWMnCTL** register is set to 1. The table below defines the effect of the event on the output signal. Value Description 0x0 Do nothing. 0x1 Invert the output signal. 0x2 Set the output signal to 0. 0x3 Set the output signal to 1.

Bit/Field	Name	Type	Reset	Description
3:2	ActLoad	R/W	0x0	Action for Counter=Load The action to be taken when the counter matches the load value. The table below defines the effect of the event on the output signal. Value Description 0x0 Do nothing. 0x1 Invert the output signal. 0x2 Set the output signal to 0. 0x3 Set the output signal to 1.
1:0	ActZero	R/W	0x0	Action for Counter=0 The action to be taken when the counter is 0. The table below defines the effect of the event on the output signal. Value Description 0x0 Do nothing. 0x1 Invert the output signal. 0x2 Set the output signal to 0. 0x3 Set the output signal to 1.

Register 51: PWM0 Dead-Band Control (PWM0DBCTL), offset 0x068

Register 52: PWM1 Dead-Band Control (PWM1DBCTL), offset 0x0A8

Register 53: PWM2 Dead-Band Control (PWM2DBCTL), offset 0x0E8

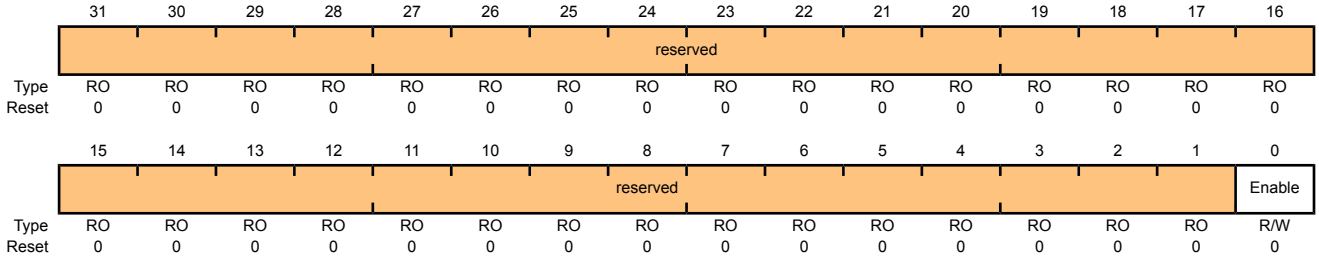
Register 54: PWM3 Dead-Band Control (PWM3DBCTL), offset 0x128

The **PWM0DBCTL** register controls the dead-band generator, which produces the PWM0 and PWM1 signals based on the PWM0A and PWM0B signals. When disabled, the PWM0A signal passes through to the PWM0 signal and the PWM0B signal passes through to the PWM1 signal. When enabled and inverting the resulting waveform, the PWM0B signal is ignored; the PWM0 signal is generated by delaying the rising edge(s) of the PWM0A signal by the value in the **PWM0DBRISE** register (see page 707), and the PWM1 signal is generated by delaying the falling edge(s) of the PWM0A signal by the value in the **PWM0DBFALL** register (see page 708). In a similar manner, PWM2 and PWM3 are produced from the PWM1A and PWM1B signals, PWM4 and PWM5 are produced from the PWM2A and PWM2B signals, and PWM6 and PWM7 are produced from the PWM3A and PWM3B signals.

If the Dead-Band Control mode is immediate (based on the DBCtlUpd field encoding in the **PWMnCTL** register), this 16-bit DBCtlUpd value is used the next time the counter reaches zero. If the update mode is synchronous, it is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 671). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWM0 Dead-Band Control (PWM0DBCTL)

Base 0x4002.8000
Offset 0x068
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
0	Enable	R/W	0	Dead-Band Generator Enable When set, the dead-band generator inserts dead bands into the output signals; when clear, it simply passes the PWM signals through.

Register 55: PWM0 Dead-Band Rising-Edge Delay (PWM0DBRISE), offset 0x06C

Register 56: PWM1 Dead-Band Rising-Edge Delay (PWM1DBRISE), offset 0x0AC

Register 57: PWM2 Dead-Band Rising-Edge Delay (PWM2DBRISE), offset 0x0EC

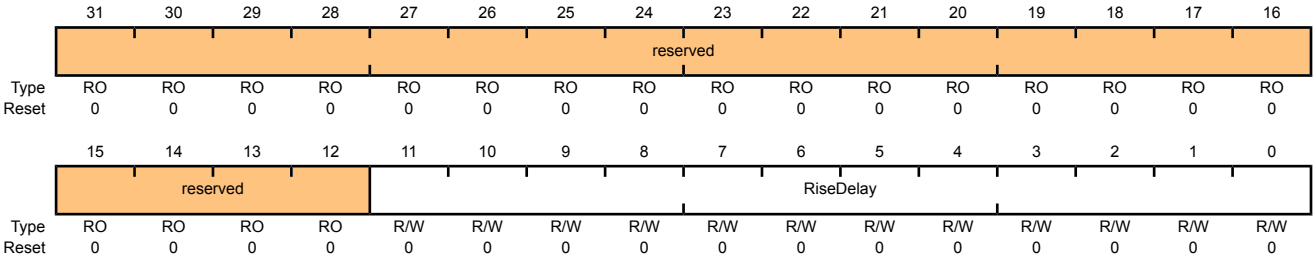
Register 58: PWM3 Dead-Band Rising-Edge Delay (PWM3DBRISE), offset 0x12C

The **PWM0DBRISE** register contains the number of clock ticks to delay the rising edge of the PWM0A signal when generating the PWM0 signal. If the dead-band generator is disabled through the **PWMnDBCTL** register, the **PWM0DBRISE** register is ignored. If the value of this register is larger than the width of a High pulse on the input PWM signal, the rising-edge delay consumes the entire High time of the signal, resulting in no High time on the output. Care must be taken to ensure that the input High time always exceeds the rising-edge delay. In a similar manner, PWM2 is generated from PWM1A with its rising edge delayed; PWM4 is produced from PWM2A with its rising edge delayed; and PWM6 is produced from PWM3A with its rising edge delayed.

If the Dead-Band Rising-Edge Delay mode is immediate (based on the DBRiseUpd field encoding in the **PWMnCTL** register), this 16-bit DBRiseUpd value is used the next time the counter reaches zero. If the update mode is synchronous, it is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 671). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWM0 Dead-Band Rising-Edge Delay (PWM0DBRISE)

Base 0x4002.8000
Offset 0x06C
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11:0	RiseDelay	R/W	0	Dead-Band Rise Delay The number of clock ticks to delay the rising edge.

Register 59: PWM0 Dead-Band Falling-Edge-Delay (PWM0DBFALL), offset 0x070

Register 60: PWM1 Dead-Band Falling-Edge-Delay (PWM1DBFALL), offset 0x0B0

Register 61: PWM2 Dead-Band Falling-Edge-Delay (PWM2DBFALL), offset 0x0F0

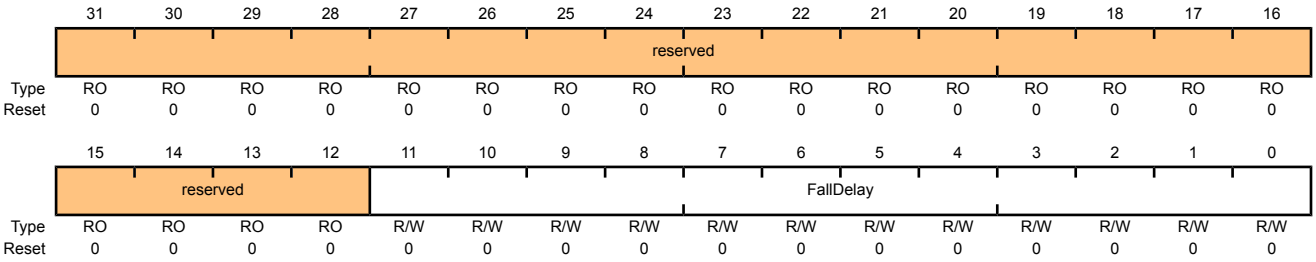
Register 62: PWM3 Dead-Band Falling-Edge-Delay (PWM3DBFALL), offset 0x130

The **PWM0DBFALL** register contains the number of clock ticks to delay the falling edge of the PWM0A signal when generating the PWM1 signal. If the dead-band generator is disabled, this register is ignored. If the value of this register is larger than the width of a Low pulse on the input PWM signal, the falling-edge delay consumes the entire Low time of the signal, resulting in no Low time on the output. Care must be taken to ensure that the input Low time always exceeds the falling-edge delay. In a similar manner, PWM3 is generated from PWM1A with its falling edge delayed, PWM5 is produced from PWM2A with its falling edge delayed, and PWM7 is produced from PWM3A with its falling edge delayed.

If the Dead-Band Falling-Edge-Delay mode is immediate (based on the DBFallUp field encoding in the **PWMnCTL** register), this 16-bit DBFallUp value is used the next time the counter reaches zero. If the update mode is synchronous, it is used the next time the counter reaches zero after a synchronous update has been requested through the **PWM Master Control (PWMCTL)** register (see page 671). If this register is rewritten before the actual update occurs, the previous value is never used and is lost.

PWM0 Dead-Band Falling-Edge-Delay (PWM0DBFALL)

Base 0x4002.8000
Offset 0x070
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
11:0	FallDelay	R/W	0x00	Dead-Band Fall Delay The number of clock ticks to delay the falling edge.

Register 63: PWM0 Fault Source 0 (PWM0FLTSRC0), offset 0x074

Register 64: PWM1 Fault Source 0 (PWM1FLTSRC0), offset 0x0B4

Register 65: PWM2 Fault Source 0 (PWM2FLTSRC0), offset 0x0F4

Register 66: PWM3 Fault Source 0 (PWM3FLTSRC0), offset 0x134

This register specifies which fault pin inputs are used to indicate a fault condition. Each bit in the following register indicates whether the corresponding fault pin is included in the fault condition. All enabled fault pins are ORed together to form the **PWMnFLTSRC0** portion of the fault condition. The **PWMnFLTSRC0** fault condition is then ORed with the **PWMnFLTSRC1** fault condition to generate the final fault condition for the PWM generator.

If the FLTSRC bit in the **PWMnCTL** register (see page 687) is clear, only the PWM Fault0 pin affects the fault condition generated. Otherwise, sources defined in **PWMnFLTSRC0** and **PWMnFLTSRC1** affect the fault condition generated.

PWM0 Fault Source 0 (PWM0FLTSRC0)

Base 0x4002.8000
Offset 0x074
Type R/W, reset 0x0000.0000

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												FAULT3	FAULT2	FAULT1	FAULT0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	FAULT3	R/W	0	Fault3 The same function as Fault0, except applied for the FAULT3 input. **Note:** The FLTSRC bit in the **PWMnCTL** register must be set for this bit to affect fault condition generation.
2	FAULT2	R/W	0	Fault2 The same function as Fault0, except applied for the FAULT2 input. **Note:** The FLTSRC bit in the **PWMnCTL** register must be set for this bit to affect fault condition generation.
1	FAULT1	R/W	0	Fault1 The same function as Fault0, except applied for the FAULT1 input. **Note:** The FLTSRC bit in the **PWMnCTL** register must be set for this bit to affect fault condition generation.

Bit/Field	Name	Type	Reset	Description
0	FAULT0	R/W	0	Fault0 Specifies the contribution of the FAULT0 input to the generation of a fault condition. Value Description 0 Suppressed The FAULT0 signal is suppressed and cannot generate a fault condition. 1 Generated The FAULT0 signal value is ORed with all other fault condition generation inputs (Fault signals).

Register 67: PWM0 Minimum Fault Period (PWM0MINFLTPER), offset 0x07C

Register 68: PWM1 Minimum Fault Period (PWM1MINFLTPER), offset 0x0BC

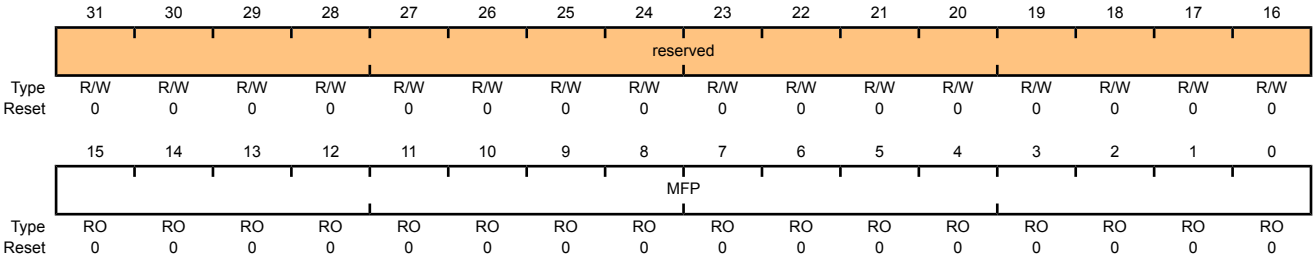
Register 69: PWM2 Minimum Fault Period (PWM2MINFLTPER), offset 0x0FC

Register 70: PWM3 Minimum Fault Period (PWM3MINFLTPER), offset 0x13C

If the MINFLTPER bit in the **PWMnCTL** register is set, this register specifies the 16-bit time-extension value to be used in extending the fault condition. The value is loaded into a 16-bit down counter, and the counter value is used to extend the fault condition. The fault condition is released in the clock immediately after the counter value reaches 0. The fault condition is asynchronous to the PWM clock; and the delay value is the product of the PWM clock period and the (MFP field value + 1) or (MFP field value + 2) depending on when the fault condition asserts with respect to the PWM clock. The counter decrements at the PWM clock rate, without pause or condition.

PWM0 Minimum Fault Period (PWM0MINFLTPER)

Base 0x4002.8000
Offset 0x07C
Type R/W, reset 0x0000.0000



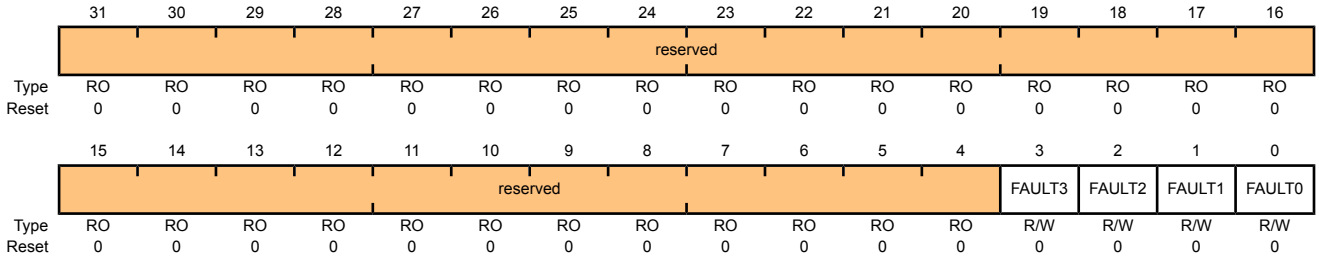
Bit/Field	Name	Type	Reset	Description
31:16	reserved	R/W	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
15:0	MFP	RO	0	Minimum Fault Period The number of PWM clocks by which a fault condition is extended when the delay is enabled by **PWMnCTL** MINFLTPER.

Register 71: PWM0 Fault Pin Logic Sense (PWM0FLTSEN), offset 0x800
Register 72: PWM1 Fault Pin Logic Sense (PWM1FLTSEN), offset 0x880
Register 73: PWM2 Fault Pin Logic Sense (PWM2FLTSEN), offset 0x900
Register 74: PWM3 Fault Pin Logic Sense (PWM3FLTSEN), offset 0x980

This register defines the PWM fault pin logic sense.

PWM0 Fault Pin Logic Sense (PWM0FLTSEN)

Base 0x4002.8000
Offset 0x800
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	FAULT3	R/W	0	Fault3 Sense The same function as FLT0SEN, except applied for the FAULT3 input.
2	FAULT2	R/W	0	Fault2 Sense The same function as FLT0SEN, except applied for the FAULT2 input.
1	FAULT1	R/W	0	Fault1 Sense The same function as FLT0SEN, except applied for the FAULT1 input.
0	FAULT0	R/W	0	Fault0 Sense This bit specifies the sense of the FAULT0 input pin, and it determines what sense is considered asserted, that is, the sense of the input (High or Low) that indicates error. Value Description 0 High 1 Low The fault sense is used to translate the incoming FAULT0 pin signal sense to an internal positive signal.

Register 75: PWM0 Fault Status 0 (PWM0FLTSTAT0), offset 0x804

Register 76: PWM1 Fault Status 0 (PWM1FLTSTAT0), offset 0x884

Register 77: PWM2 Fault Status 0 (PWM2FLTSTAT0), offset 0x904

Register 78: PWM3 Fault Status 0 (PWM3FLTSTAT0), offset 0x984

Along with the **PWMnFLTSTAT1** register, this register provides status regarding the fault condition inputs.

If the LATCH bit in the **PWMnCTL** register is clear, the contents of the **PWMnFLTSTAT0** register are read-only (RO) and provide the current state of the FAULTn inputs.

If the LATCH bit in the **PWMnCTL** register is set, the contents of the **PWMnFLTSTAT0** register are read / write 1 to clear (R/W1C) and provide a latched version of the FAULTn inputs. In this mode, the register bits are cleared by writing a 1 to a set bit. The FAULTn inputs are recorded after their sense is adjusted in the generator.

The contents of this register can only be written if the fault source extensions are enabled (the FLTSRC bit in the **PWMnCTL** register is set).

PWM0 Fault Status 0 (PWM0FLTSTAT0)

Base 0x4002.8000
Offset 0x804
Type -, reset 0x0000.0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved												FAULT3	FAULT2	FAULT1	FAULT0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	-	-	-	-
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	FAULT3	-	0	Fault Input 3 The same function as FAULT0, except applied for the FAULT3 input.
2	FAULT2	-	0	Fault Input 2 The same function as FAULT0, except applied for the FAULT2 input.
1	FAULT1	-	0	Fault Input 1 The same function as FAULT0, except applied for the FAULT1 input.

Bit/Field	Name	Type	Reset	Description
0	FAULT0	-	0	Fault Input 0 If the **PWMnCTL** register LATCH bit is clear, this bit is RO and represents the current state of the FAULT0 input signal after the logic sense adjustment. If the **PWMnCTL** register LATCH bit is set, this bit is R/W1C and represents a sticky version of the FAULT0 input signal after the logic sense adjustment. ■ If FAULT0 is set, the input transitioned to the active state previously. ■ If FAULT0 is clear, the input has not transitioned to the active state since the last time it was cleared. ■ The FAULT0 bit is cleared by writing it with the value 1.

21 Quadrature Encoder Interface (QEI)

A quadrature encoder, also known as a 2-channel incremental encoder, converts linear displacement into a pulse signal. By monitoring both the number of pulses and the relative phase of the two signals, you can track the position, direction of rotation, and speed. In addition, a third channel, or index signal, can be used to reset the position counter.

The Stellaris® quadrature encoder interface (QEI) module interprets the code produced by a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, it can capture a running estimate of the velocity of the encoder wheel.

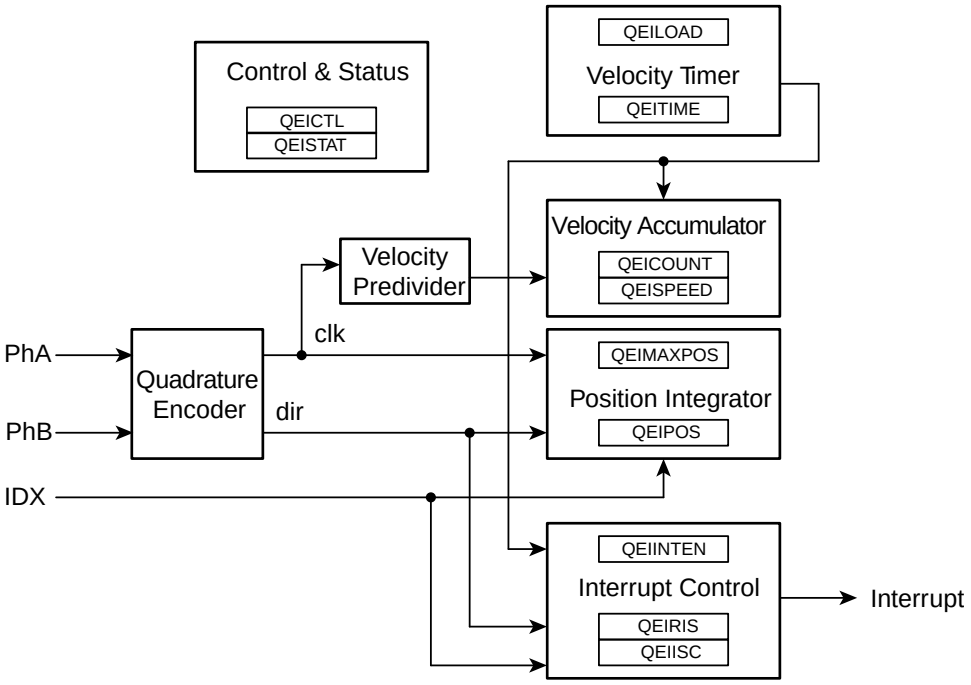
The Stellaris® quadrature encoder has the following features:

- Position integrator that tracks the encoder position
- Velocity capture using built-in timer
- The input frequency of the QEI inputs may be as high as 1/4 of the processor frequency (for example, 12.5 MHz for a 50-MHz system)
- Interrupt generation on:
  - Index pulse
  - Velocity-timer expiration
  - Direction change
  - Quadrature error detection

21.1 Block Diagram

Figure 21-1 on page 716 provides a block diagram of a Stellaris® QEI module.

Figure 21-1. QEI Block Diagram



21.2 Functional Description

The QEI module interprets the two-bit gray code produced by a quadrature encoder wheel to integrate position over time and determine direction of rotation. In addition, it can capture a running estimate of the velocity of the encoder wheel.

The position integrator and velocity capture can be independently enabled, though the position integrator must be enabled before the velocity capture can be enabled. The two phase signals, PhA and PhB, can be swapped before being interpreted by the QEI module to change the meaning of forward and backward, and to correct for miswiring of the system. Alternatively, the phase signals can be interpreted as a clock and direction signal as output by some encoders.

The QEI module supports two modes of signal operation: quadrature phase mode and clock/direction mode. In quadrature phase mode, the encoder produces two clocks that are 90 degrees out of phase; the edge relationship is used to determine the direction of rotation. In clock/direction mode, the encoder produces a clock signal to indicate steps and a direction signal to indicate the direction of rotation. This mode is determined by the SigMode bit of the **QEI Control (QEICTL)** register (see page 720).

When the QEI module is set to use the quadrature phase mode (SigMode bit equals zero), the capture mode for the position integrator can be set to update the position counter on every edge of the PhA signal or to update on every edge of both PhA and PhB. Updating the position counter on every PhA and PhB provides more positional resolution at the cost of less range in the positional counter.

When edges on PhA lead edges on PhB , the position counter is incremented. When edges on PhB lead edges on PhA , the position counter is decremented. When a rising and falling edge pair is seen on one of the phases without any edges on the other, the direction of rotation has changed.

The positional counter is automatically reset on one of two conditions: sensing the index pulse or reaching the maximum position value. Which mode is determined by the ResMode bit of the **QEI Control (QEICTL)** register.

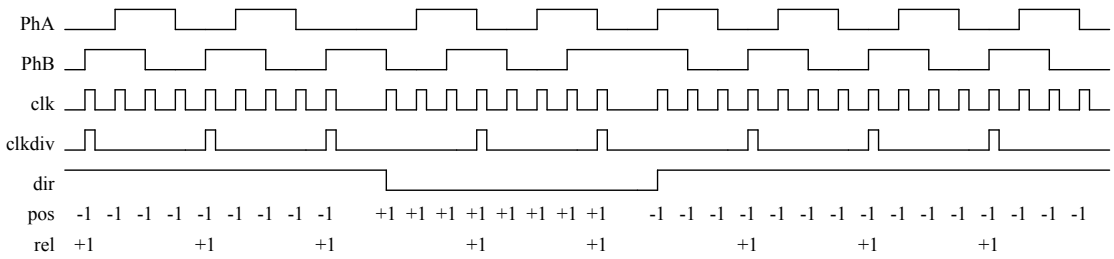
When ResMode is 0, the positional counter is reset when the index pulse is sensed. This limits the positional counter to the values [0:N-1], where N is the number of phase edges in a full revolution of the encoder wheel. The **QEIMAXPOS** register must be programmed with N-1 so that the reverse direction from position 0 can move the position counter to N-1. In this mode, the position register contains the absolute position of the encoder relative to the index (or home) position once an index pulse has been seen.

When ResMode is 1, the positional counter is constrained to the range [0:M], where M is the programmable maximum value. The index pulse is ignored by the positional counter in this mode.

The velocity capture has a configurable timer and a count register. It counts the number of phase edges (using the same configuration as for the position integrator) in a given time period. The edge count from the previous time period is available to the controller via the **QEISPEED** register, while the edge count for the current time period is being accumulated in the **QEICOUNT** register. As soon as the current time period is complete, the total number of edges counted in that time period is made available in the **QEISPEED** register (losing the previous value), the **QEICOUNT** is reset to 0, and counting commences on a new time period. The number of edges counted in a given time period is directly proportional to the velocity of the encoder.

Figure 21-2 on page 717 shows how the Stellaris® quadrature encoder converts the phase input signals into clock pulses, the direction signal, and how the velocity predivider operates (in Divide by 4 mode).

Figure 21-2. Quadrature Encoder and Velocity Predivider Operation



The period of the timer is configurable by specifying the load value for the timer in the **QEILOAD** register. When the timer reaches zero, an interrupt can be triggered, and the hardware reloads the timer with the **QEILOAD** value and continues to count down. At lower encoder speeds, a longer timer period is needed to be able to capture enough edges to have a meaningful result. At higher encoder speeds, both a shorter timer period and/or the velocity predivider can be used.

The following equation converts the velocity counter value into an rpm value:

```
rpm = (clock * (2 ^ VelDiv) * Speed * 60) ÷ (Load * ppr * edges)
```

where:

clock is the controller clock rate

ppr is the number of pulses per revolution of the physical encoder

edges is 2 or 4, based on the capture mode set in the **QEICTL** register (2 for CapMode set to 0 and 4 for CapMode set to 1)

For example, consider a motor running at 600 rpm. A 2048 pulse per revolution quadrature encoder is attached to the motor, producing 8192 phase edges per revolution. With a velocity predivider of ÷1 (VelDiv set to 0) and clocking on both PhA and PhB edges, this results in 81,920 pulses per second (the motor turns 10 times per second). If the timer were clocked at 10,000 Hz, and the load value was 2,500 (¼ of a second), it would count 20,480 pulses per update. Using the above equation:

```
rpm = (10000 * 1 * 20480 * 60) ÷ (2500 * 2048 * 4) = 600 rpm
```

Now, consider that the motor is sped up to 3000 rpm. This results in 409,600 pulses per second, or 102,400 every ¼ of a second. Again, the above equation gives:

```
rpm = (10000 * 1 * 102400 * 60) ÷ (2500 * 2048 * 4) = 3000 rpm
```

Care must be taken when evaluating this equation since intermediate values may exceed the capacity of a 32-bit integer. In the above examples, the clock is 10,000 and the divider is 2,500; both could be predivided by 100 (at compile time if they are constants) and therefore be 100 and 25. In fact, if they were compile-time constants, they could also be reduced to a simple multiply by 4, cancelled by the ÷4 for the edge-count factor.

Important: Reducing constant factors at compile time is the best way to control the intermediate values of this equation, as well as reducing the processing requirement of computing this equation.

The division can be avoided by selecting a timer load value such that the divisor is a power of 2; a simple shift can therefore be done in place of the division. For encoders with a power of 2 pulses per revolution, this is a simple matter of selecting a power of 2 load value. For other encoders, a load value must be selected such that the product is very close to a power of two. For example, a 100 pulse per revolution encoder could use a load value of 82, resulting in 32,800 as the divisor, which is 0.09% above 2^{14}; in this case a shift by 15 would be an adequate approximation of the divide in most cases. If absolute accuracy were required, the controller's divide instruction could be used.

The QEI module can produce a controller interrupt on several events: phase error, direction change, reception of the index pulse, and expiration of the velocity timer. Standard masking, raw interrupt status, interrupt status, and interrupt clear capabilities are provided.

21.3 Initialization and Configuration

The following example shows how to configure the Quadrature Encoder module to read back an absolute position:

1. Enable the QEI clock by writing a value of 0x0000.0100 to the **RCGC1** register in the System Control module.
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.
4. Configure the quadrature encoder to capture edges on both signals and maintain an absolute position by resetting on index pulses. Using a 1000-line encoder at four edges per line, there are 4000 pulses per revolution; therefore, set the maximum position to 3999 (0xF9F) since the count is zero-based.

- Write the **QEICTL** register with the value of 0x0000.0018.
- Write the **QEIMAXPOS** register with the value of 0x0000.0F9F.

5. Enable the quadrature encoder by setting bit 0 of the **QEICTL** register.
6. Delay for some time.
7. Read the encoder position by reading the **QEIPOS** register value.

21.4 Register Map

Table 21-1 on page 719 lists the QEI registers. The offset listed is a hexadecimal increment to the register's address, relative to the module's base address:

- QEI0: 0x4002.C000

Table 21-1. QEI Register Map

Offset	Name	Type	Reset	Description	See page
0x000	QEICTL	R/W	0x0000.0000	QEI Control	720
0x004	QEISTAT	RO	0x0000.0000	QEI Status	722
0x008	QEIPOS	R/W	0x0000.0000	QEI Position	723
0x00C	QEIMAXPOS	R/W	0x0000.0000	QEI Maximum Position	724
0x010	QEILOAD	R/W	0x0000.0000	QEI Timer Load	725
0x014	QEITIME	RO	0x0000.0000	QEI Timer	726
0x018	QEICOUNT	RO	0x0000.0000	QEI Velocity Counter	727
0x01C	QEISPEED	RO	0x0000.0000	QEI Velocity	728
0x020	QEIINTEN	R/W	0x0000.0000	QEI Interrupt Enable	729
0x024	QEIRIS	RO	0x0000.0000	QEI Raw Interrupt Status	730
0x028	QEIISC	R/W1C	0x0000.0000	QEI Interrupt Status and Clear	731

21.5 Register Descriptions

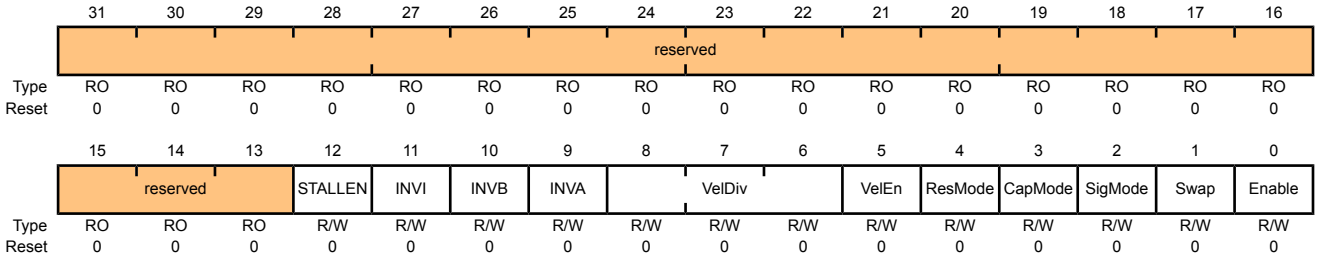
The remainder of this section lists and describes the QEI registers, in numerical order by address offset.

Register 1: QEI Control (QEICTL), offset 0x000

This register contains the configuration of the QEI module. Separate enables are provided for the quadrature encoder and the velocity capture blocks; the quadrature encoder must be enabled in order to capture the velocity, but the velocity does not need to be captured in applications that do not need it. The phase signal interpretation, phase swap, Position Update mode, Position Reset mode, and velocity predivider are all set via this register.

QEI Control (QEICTL)

QEI0 base: 0x4002.C000
Offset 0x000
Type R/W, reset 0x0000.0000



Bit/Field	Name	Type	Reset	Description
31:13	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
12	STALLEN	R/W	0	Stall QEI When set, the QEI stalls when the microcontroller asserts Halt.
11	INVI	R/W	0	Invert Index Pulse When set , the input Index Pulse is inverted.
10	INVB	R/W	0	Invert PhB When set, the PhB input is inverted.
9	INVA	R/W	0	Invert PhA When set, the PhA input is inverted.
8:6	VelDiv	R/W	0x0	Predivide Velocity A predivider of the input quadrature pulses before being applied to the QEICOUNT accumulator. This field can be set to the following values: Value Predivider 0x0 ÷1 0x1 ÷2 0x2 ÷4 0x3 ÷8 0x4 ÷16 0x5 ÷32 0x6 ÷64 0x7 ÷128

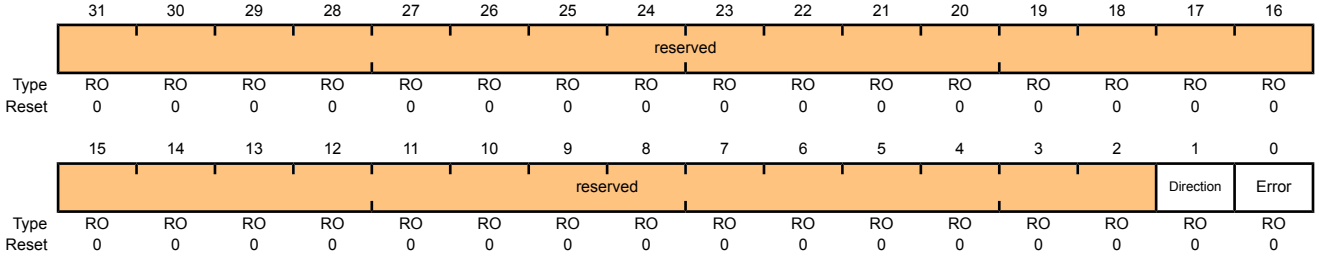
Bit/Field	Name	Type	Reset	Description
5	VelEn	R/W	0	Capture Velocity When set, enables capture of the velocity of the quadrature encoder.
4	ResMode	R/W	0	Reset Mode The Reset mode for the position counter. When 0, the position counter is reset when it reaches the maximum; when 1, the position counter is reset when the index pulse is captured.
3	CapMode	R/W	0	Capture Mode The Capture mode defines the phase edges that are counted in the position. When 0, only the PhA edges are counted; when 1, the PhA and PhB edges are counted, providing twice the positional resolution but half the range.
2	SigMode	R/W	0	Signal Mode When 1, the PhA and PhB signals are clock and direction; when 0, they are quadrature phase signals.
1	Swap	R/W	0	Swap Signals Swaps the PhA and PhB signals.
0	Enable	R/W	0	Enable QEI Enables the quadrature encoder module.

Register 2: QEI Status (QEISTAT), offset 0x004

This register provides status about the operation of the QEI module.

QEI Status (QEISTAT)

QEI0 base: 0x4002.C000
Offset 0x004
Type RO, reset 0x0000.0000



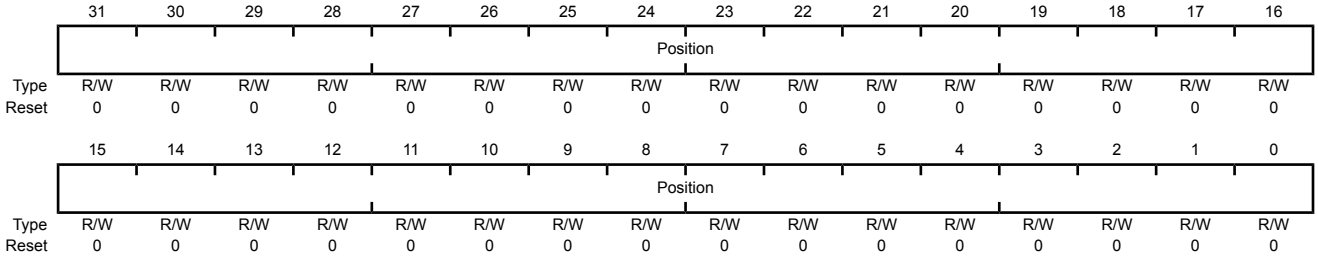
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
1	Direction	RO	0	Direction of Rotation Indicates the direction the encoder is rotating. The `Direction` values are defined as follows: Value Description 0 Forward rotation 1 Reverse rotation
0	Error	RO	0	Error Detected Indicates that an error was detected in the gray code sequence (that is, both signals changing at the same time).

Register 3: QEI Position (QEIPOS), offset 0x008

This register contains the current value of the position integrator. Its value is updated by inputs on the QEI phase inputs, and can be set to a specific value by writing to it.

QEI Position (QEIPOS)

QEI0 base: 0x4002.C000
Offset 0x008
Type R/W, reset 0x0000.0000



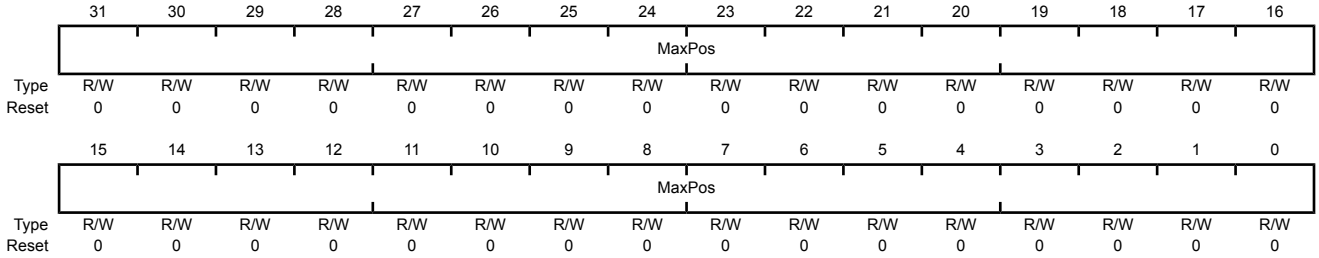
Bit/Field	Name	Type	Reset	Description
31:0	Position	R/W	0x00	Current Position Integrator Value The current value of the position integrator.

Register 4: QEI Maximum Position (QEIMAXPOS), offset 0x00C

This register contains the maximum value of the position integrator. When moving forward, the position register resets to zero when it increments past this value. When moving backward, the position register resets to this value when it decrements from zero.

QEI Maximum Position (QEIMAXPOS)

QEI0 base: 0x4002.C000
Offset 0x00C
Type R/W, reset 0x0000.0000



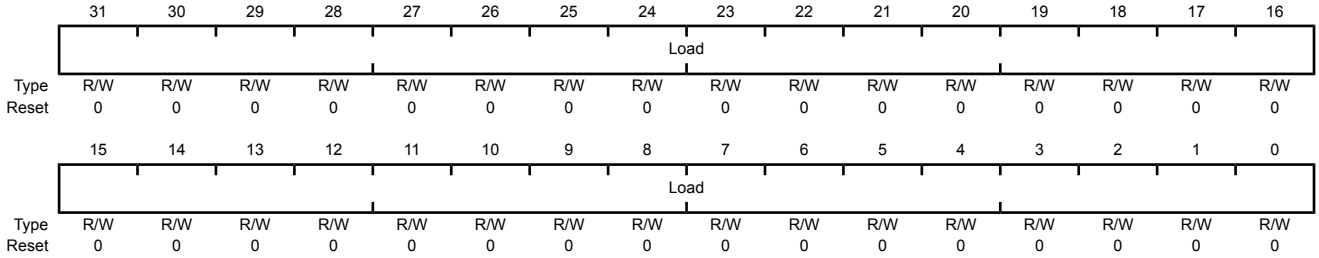
Bit/Field	Name	Type	Reset	Description
31:0	MaxPos	R/W	0x00	Maximum Position Integrator Value The maximum value of the position integrator.

Register 5: QEI Timer Load (QEILOAD), offset 0x010

This register contains the load value for the velocity timer. Since this value is loaded into the timer the clock cycle after the timer is zero, this value should be one less than the number of clocks in the desired period. So, for example, to have 2000 clocks per timer period, this register should contain 1999.

QEI Timer Load (QEILOAD)

QEI0 base: 0x4002.C000
Offset 0x010
Type R/W, reset 0x0000.0000



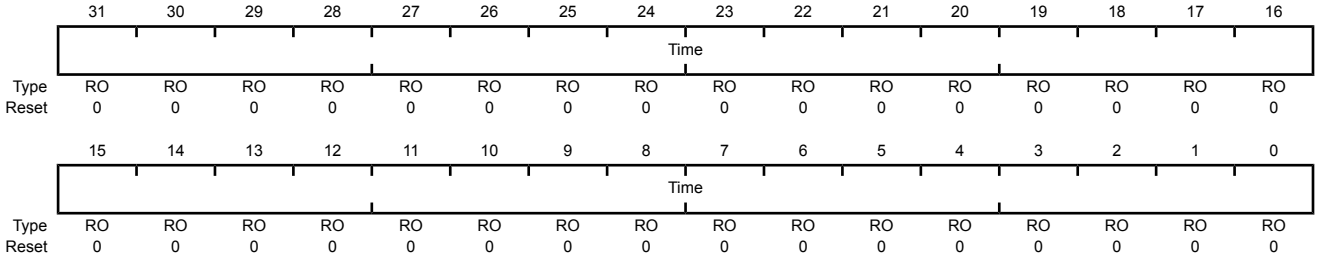
Bit/Field	Name	Type	Reset	Description
31:0	Load	R/W	0x00	Velocity Timer Load Value The load value for the velocity timer.

Register 6: QEI Timer (QEITIME), offset 0x014

This register contains the current value of the velocity timer. This counter does not increment when VelEn in **QEICTL** is 0.

QEI Timer (QEITIME)

QEI0 base: 0x4002.C000
Offset 0x014
Type RO, reset 0x0000.0000



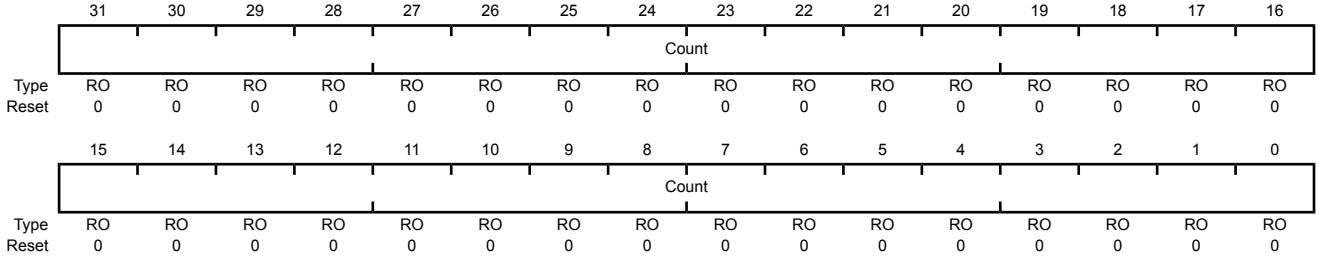
Bit/Field	Name	Type	Reset	Description
31:0	Time	RO	0x00	Velocity Timer Current Value The current value of the velocity timer.

Register 7: QEI Velocity Counter (QEICOUNT), offset 0x018

This register contains the running count of velocity pulses for the current time period. Since this is a running total, the time period to which it applies cannot be known with precision (that is, a read of this register does not necessarily correspond to the time returned by the **QEITIME** register since there is a small window of time between the two reads, during which time either value may have changed). The **QEISPEED** register should be used to determine the actual encoder velocity; this register is provided for information purposes only. This counter does not increment when VelEn in **QEICTL** is 0.

QEI Velocity Counter (QEICOUNT)

QEI0 base: 0x4002.C000
Offset 0x018
Type RO, reset 0x0000.0000



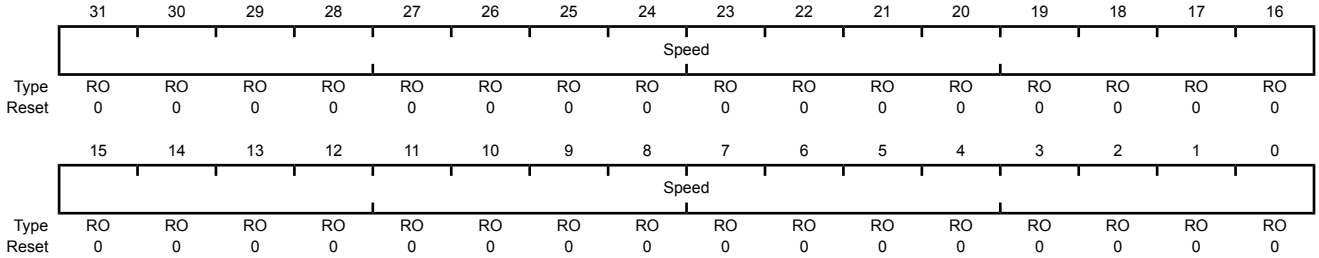
Bit/Field	Name	Type	Reset	Description
31:0	Count	RO	0x00	Velocity Pulse Count The running total of encoder pulses during this velocity timer period.

Register 8: QEI Velocity (QEISPEED), offset 0x01C

This register contains the most recently measured velocity of the quadrature encoder. This corresponds to the number of velocity pulses counted in the previous velocity timer period. This register does not update when VelEn in **QEICTL** is 0.

QEI Velocity (QEISPEED)

QEI0 base: 0x4002.C000
Offset 0x01C
Type RO, reset 0x0000.0000



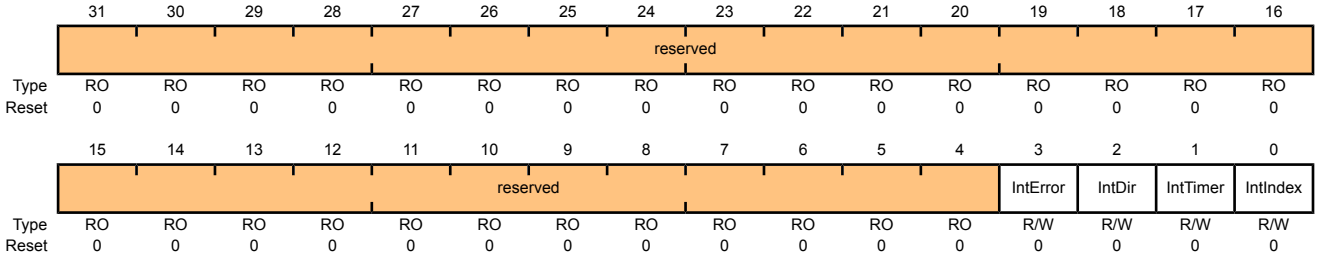
Bit/Field	Name	Type	Reset	Description
31:0	Speed	RO	0x00	Velocity The measured speed of the quadrature encoder in pulses per period.

Register 9: QEI Interrupt Enable (QEIINTEN), offset 0x020

This register contains enables for each of the QEI module's interrupts. An interrupt is asserted to the controller if its corresponding bit in this register is set to 1.

QEI Interrupt Enable (QEIINTEN)

QEI0 base: 0x4002.C000
Offset 0x020
Type R/W, reset 0x0000.0000



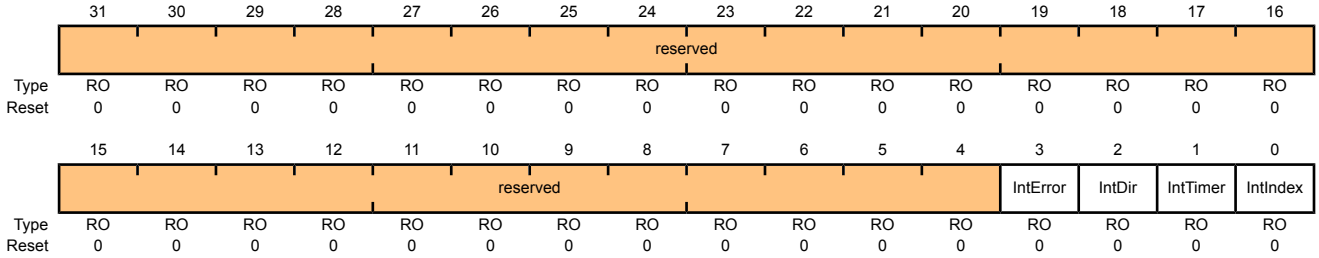
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IntError	R/W	0	Phase Error Interrupt Enable When 1, an interrupt occurs when a phase error is detected.
2	IntDir	R/W	0	Direction Change Interrupt Enable When 1, an interrupt occurs when the direction changes.
1	IntTimer	R/W	0	Timer Expires Interrupt Enable When 1, an interrupt occurs when the velocity timer expires.
0	IntIndex	R/W	0	Index Pulse Detected Interrupt Enable When 1, an interrupt occurs when the index pulse is detected.

Register 10: QEI Raw Interrupt Status (QEIRIS), offset 0x024

This register provides the current set of interrupt sources that are asserted, regardless of whether they cause an interrupt to be asserted to the controller (this is set through the **QEIINTEN** register). Bits set to 1 indicate the latched events that have occurred; a zero bit indicates that the event in question has not occurred.

QEI Raw Interrupt Status (QEIRIS)

QEI0 base: 0x4002.C000
Offset 0x024
Type RO, reset 0x0000.0000



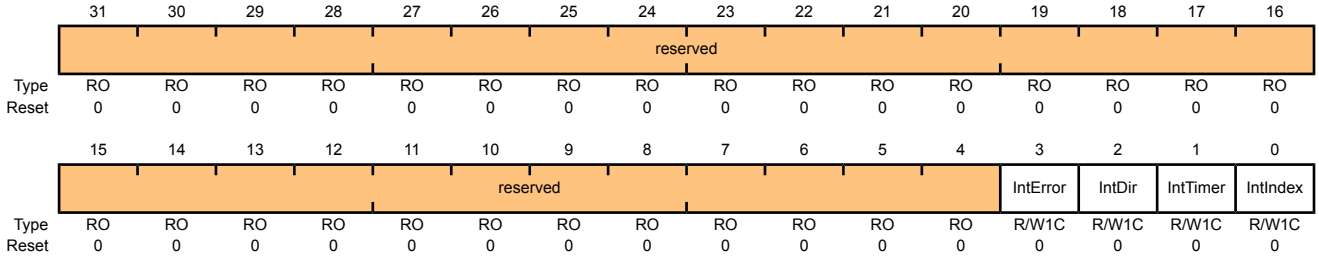
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IntError	RO	0	Phase Error Detected Indicates that a phase error was detected.
2	IntDir	RO	0	Direction Change Detected Indicates that the direction has changed.
1	IntTimer	RO	0	Velocity Timer Expired Indicates that the velocity timer has expired.
0	IntIndex	RO	0	Index Pulse Asserted Indicates that the index pulse has occurred.

Register 11: QEI Interrupt Status and Clear (QEIISC), offset 0x028

This register provides the current set of interrupt sources that are asserted to the controller. Bits set to 1 indicate the latched events that have occurred; a zero bit indicates that the event in question has not occurred. This is a R/W1C register; writing a 1 to a bit position clears the corresponding interrupt reason.

QEI Interrupt Status and Clear (QEIISC)

QEI0 base: 0x4002.C000
Offset 0x028
Type R/W1C, reset 0x0000.0000

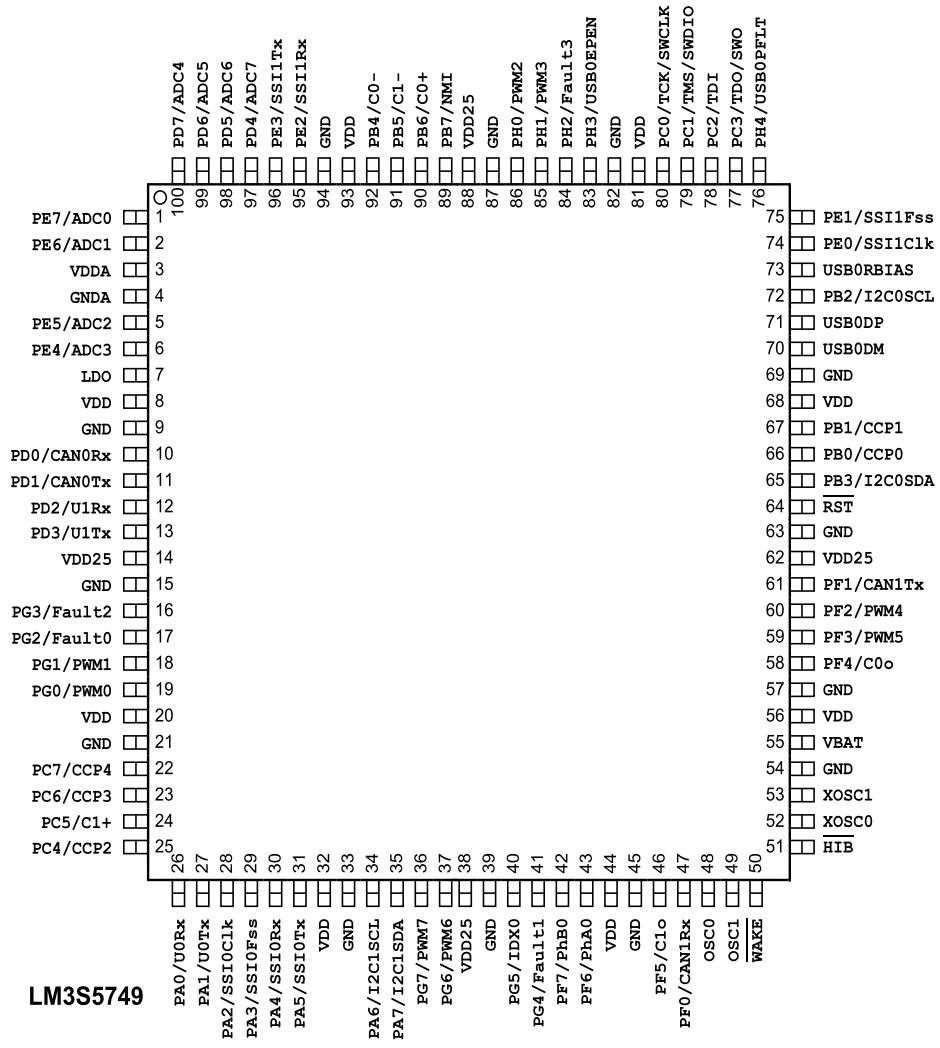


Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0x00	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
3	IntError	R/W1C	0	Phase Error Interrupt Indicates that a phase error was detected.
2	IntDir	R/W1C	0	Direction Change Interrupt Indicates that the direction has changed.
1	IntTimer	R/W1C	0	Velocity Timer Expired Interrupt Indicates that the velocity timer has expired.
0	IntIndex	R/W1C	0	Index Pulse Interrupt Indicates that the index pulse has occurred.

22 Pin Diagram

The LM3S5749 microcontroller pin diagram is shown below.

Figure 22-1. 100-Pin LQFP Package Pin Diagram



23 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 four JTAG pins (PC[3:0]) which default to the JTAG functionality.

Table 23-1 on page 733 shows the pin-to-signal-name mapping, including functional characteristics of the signals. Table 23-2 on page 738 lists the signals in alphabetical order by signal name.

Table 23-3 on page 742 groups the signals by functionality, except for GPIOs. Table 23-4 on page 745 lists the GPIO pins and their alternate functionality.

Table 23-1. Signals by Pin Number

Pin Number	Pin Name	Pin Type	Buffer Type	Description
1	PE7	I/O	Analog	GPIO port E bit 7
	ADC0	I	Analog	ADC 0 input
2	PE6	I/O	Analog	GPIO port E bit 6
	ADC1	I	Analog	ADC 1 input
3	VDDA	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
4	GNDA	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
5	PE5	I/O	Analog	GPIO port E bit 5
	ADC2	I	Analog	ADC 2 input
6	PE4	I/O	Analog	GPIO port E bit 4
	ADC3	I	Analog	ADC 3 input
7	LDO	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. The LDO pin must also be connected to the VDD25 pins at the board level in addition to the decoupling capacitor(s).
8	VDD	-	Power	Positive supply for I/O and some logic.
9	GND	-	Power	Ground reference for logic and I/O pins.
10	PD0	I/O	TTL	GPIO port D bit 0
	CAN0Rx	I	TTL	CAN module 0 receive
11	PD1	I/O	TTL	GPIO port D bit 1
	CAN0Tx	O	TTL	CAN module 0 transmit
12	PD2	I/O	TTL	GPIO port D bit 2
	U1Rx	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.

Pin Number	Pin Name	Pin Type	Buffer Type	Description
13	PD3	I/O	TTL	GPIO port D bit 3
	U1Tx	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
14	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
15	GND	-	Power	Ground reference for logic and I/O pins.
16	PG3	I/O	TTL	GPIO port G bit 3
	Fault2	I	TTL	PWM Fault 2
17	PG2	I/O	TTL	GPIO port G bit 2
	Fault0	I	TTL	PWM Fault 0
18	PG1	I/O	TTL	GPIO port G bit 1
	PWM1	O	TTL	PWM 1
19	PG0	I/O	TTL	GPIO port G bit 0
	PWM0	O	TTL	PWM 0
20	VDD	-	Power	Positive supply for I/O and some logic.
21	GND	-	Power	Ground reference for logic and I/O pins.
22	PC7	I/O	TTL	GPIO port C bit 7
	CCP4	I/O	TTL	Capture/Compare/PWM 4
23	PC6	I/O	TTL	GPIO port C bit 6
	CCP3	I/O	TTL	Capture/Compare/PWM 3
24	PC5	I/O	TTL	GPIO port C bit 5
	C1+	I	Analog	Analog comparator positive input
25	PC4	I/O	TTL	GPIO port C bit 4
	CCP2	I/O	TTL	Capture/Compare/PWM 2
26	PA0	I/O	TTL	GPIO port A bit 0
	U0Rx	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
27	PA1	I/O	TTL	GPIO port A bit 1
	U0Tx	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
28	PA2	I/O	TTL	GPIO port A bit 2
	SSI0Clk	I/O	TTL	SSI module 0 clock
29	PA3	I/O	TTL	GPIO port A bit 3
	SSI0Fss	I/O	TTL	SSI module 0 frame
30	PA4	I/O	TTL	GPIO port A bit 4
	SSI0Rx	I	TTL	SSI module 0 receive
31	PA5	I/O	TTL	GPIO port A bit 5
	SSI0Tx	O	TTL	SSI module 0 transmit
32	VDD	-	Power	Positive supply for I/O and some logic.
33	GND	-	Power	Ground reference for logic and I/O pins.
34	PA6	I/O	TTL	GPIO port A bit 6
	I2C1SCL	I/O	OD	I2C module 1 clock

Pin Number	Pin Name	Pin Type	Buffer Type	Description
35	PA7	I/O	TTL	GPIO port A bit 7
	I2C1SDA	I/O	OD	I2C module 1 data
36	PG7	I/O	TTL	GPIO port G bit 7
	PWM7	O	TTL	PWM 7
37	PG6	I/O	TTL	GPIO port G bit 6
	PWM6	O	TTL	PWM 6
38	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
39	GND	-	Power	Ground reference for logic and I/O pins.
40	PG5	I/O	TTL	GPIO port G bit 5
	IDX0	I	TTL	QEI module 0 index
41	PG4	I/O	TTL	GPIO port G bit 4
	Fault1	I	TTL	PWM Fault 1
42	PF7	I/O	TTL	GPIO port F bit 7
	PhB0	I	TTL	QEI module 0 Phase B
43	PF6	I/O	TTL	GPIO port F bit 6
	PhA0	I	TTL	QEI module 0 Phase A
44	VDD	-	Power	Positive supply for I/O and some logic.
45	GND	-	Power	Ground reference for logic and I/O pins.
46	PF5	I/O	TTL	GPIO port F bit 5
	C1o	O	TTL	Analog comparator 1 output
47	PF0	I/O	TTL	GPIO port F bit 0
	CAN1Rx	I	TTL	CAN module 1 receive
48	OSC0	I	Analog	Main oscillator crystal input or an external clock reference input.
49	OSC1	O	Analog	Main oscillator crystal output.
50	$\overline{\text{WAKE}}$	I	-	An external input that brings the processor out of hibernate mode when asserted.
51	$\overline{\text{HIB}}$	O	OD	An output that indicates the processor is in hibernate mode.
52	XOSC0	I	Analog	Hibernation Module oscillator crystal input or an external clock reference input. Note that this is either a 4.19-MHz crystal or a 32.768-kHz oscillator for the Hibernation Module RTC. See the CLKSEL bit in the **HIBCTL** register.
53	XOSC1	O	Analog	Hibernation Module oscillator crystal output.
54	GND	-	Power	Ground reference for logic and I/O pins.
55	VBAT	-	Power	Power source for the Hibernation Module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation Module power-source supply.
56	VDD	-	Power	Positive supply for I/O and some logic.
57	GND	-	Power	Ground reference for logic and I/O pins.

Pin Number	Pin Name	Pin Type	Buffer Type	Description
58	PF4	I/O	TTL	GPIO port F bit 4
	C0o	O	TTL	Analog comparator 0 output
59	PF3	I/O	TTL	GPIO port F bit 3
	PWM5	O	TTL	PWM 5
60	PF2	I/O	TTL	GPIO port F bit 2
	PWM4	O	TTL	PWM 4
61	PF1	I/O	TTL	GPIO port F bit 1
	CAN1Tx	O	TTL	CAN module 1 transmit
62	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
63	GND	-	Power	Ground reference for logic and I/O pins.
64	$\overline{\text{RST}}$	I/O	TTL	System reset input.
65	PB3	I/O	TTL	GPIO port B bit 3
	I2C0SDA	I/O	OD	I2C module 0 data
66	PB0	I/O	TTL	GPIO port B bit 0
	CCP0	I/O	TTL	Capture/Compare/PWM 0
67	PB1	I/O	TTL	GPIO port B bit 1
	CCP1	I/O	TTL	Capture/Compare/PWM 1
68	VDD	-	Power	Positive supply for I/O and some logic.
69	GND	-	Power	Ground reference for logic and I/O pins.
70	USB0DM	I/O	Analog	Bidirectional differential data pin (D- per USB specification).
71	USB0DP	I/O	Analog	Bidirectional differential data pin (D+ per USB specification).
72	PB2	I/O	TTL	GPIO port B bit 2
	I2C0SCL	I/O	OD	I2C module 0 clock
73	USB0RBIAS	I	Analog	9.1 KOhm resistor (1% precision) used internally for USB analog circuitry.
74	PE0	I/O	TTL	GPIO port E bit 0
	SSI1Clk	I/O	TTL	SSI module 1 clock
75	PE1	I/O	TTL	GPIO port E bit 1
	SSI1Fss	I/O	TTL	SSI module 1 frame
76	PH4	I/O	TTL	GPIO port H bit 4
	USB0PFLT	I	TTL	Used in Host mode by an external power source to indicate an error state by that power source.
77	PC3	I/O	TTL	GPIO port C bit 3
	TDO	O	TTL	JTAG TDO and SWO
	SWO	O	TTL	JTAG TDO and SWO
78	PC2	I/O	TTL	GPIO port C bit 2
	TDI	I	TTL	JTAG TDI
79	PC1	I/O	TTL	GPIO port C bit 1
	TMS	I/O	TTL	JTAG TMS and SWDIO
	SWDIO	I/O	TTL	JTAG TMS and SWDIO

Pin Number	Pin Name	Pin Type	Buffer Type	Description
80	PC0	I/O	TTL	GPIO port C bit 0
	TCK	I	TTL	JTAG/SWD CLK
	SWCLK	I	TTL	JTAG/SWD CLK
81	VDD	-	Power	Positive supply for I/O and some logic.
82	GND	-	Power	Ground reference for logic and I/O pins.
83	PH3	I/O	TTL	GPIO port H bit 3
	USB0EPEN	O	TTL	Used in Host mode to control an external power source to supply power to the USB bus.
84	PH2	I/O	TTL	GPIO port H bit 2
	Fault3	I	TTL	PWM Fault 3
85	PH1	I/O	TTL	GPIO port H bit 1
	PWM3	O	TTL	PWM 3
86	PH0	I/O	TTL	GPIO port H bit 0
	PWM2	O	TTL	PWM 2
87	GND	-	Power	Ground reference for logic and I/O pins.
88	VDD25	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
89	PB7	I/O	TTL	GPIO port B bit 7
	NMI	I	TTL	Non maskable interrupt
90	PB6	I/O	TTL	GPIO port B bit 6
	C0+	I	Analog	Analog comparator 0 positive input
91	PB5	I/O	TTL	GPIO port B bit 5
	C1-	I	Analog	Analog comparator 1 negative input
92	PB4	I/O	TTL	GPIO port B bit 4
	C0-	I	Analog	Analog comparator 0 negative input
93	VDD	-	Power	Positive supply for I/O and some logic.
94	GND	-	Power	Ground reference for logic and I/O pins.
95	PE2	I/O	TTL	GPIO port E bit 2
	SSI1Rx	I	TTL	SSI module 1 receive
96	PE3	I/O	TTL	GPIO port E bit 3
	SSI1Tx	O	TTL	SSI module 1 transmit
97	PD4	I/O	Analog	GPIO port D bit 4
	ADC7	I	Analog	ADC 7 input
98	PD5	I/O	Analog	GPIO port D bit 5
	ADC6	I	Analog	ADC 6 input
99	PD6	I/O	Analog	GPIO port D bit 6
	ADC5	I	Analog	ADC 5 input
100	PD7	I/O	Analog	GPIO port D bit 7
	ADC4	I	Analog	ADC 4 input

Table 23-2. Signals by Signal Name

Pin Name	Pin Number	Pin Type	Buffer Type	Description
ADC0	1	I	Analog	ADC 0 input
ADC1	2	I	Analog	ADC 1 input
ADC2	5	I	Analog	ADC 2 input
ADC3	6	I	Analog	ADC 3 input
ADC4	100	I	Analog	ADC 4 input
ADC5	99	I	Analog	ADC 5 input
ADC6	98	I	Analog	ADC 6 input
ADC7	97	I	Analog	ADC 7 input
C0+	90	I	Analog	Analog comparator 0 positive input
C0-	92	I	Analog	Analog comparator 0 negative input
C0o	58	O	TTL	Analog comparator 0 output
C1+	24	I	Analog	Analog comparator positive input
C1-	91	I	Analog	Analog comparator 1 negative input
C1o	46	O	TTL	Analog comparator 1 output
CAN0Rx	10	I	TTL	CAN module 0 receive
CAN0Tx	11	O	TTL	CAN module 0 transmit
CAN1Rx	47	I	TTL	CAN module 1 receive
CAN1Tx	61	O	TTL	CAN module 1 transmit
CCP0	66	I/O	TTL	Capture/Compare/PWM 0
CCP1	67	I/O	TTL	Capture/Compare/PWM 1
CCP2	25	I/O	TTL	Capture/Compare/PWM 2
CCP3	23	I/O	TTL	Capture/Compare/PWM 3
CCP4	22	I/O	TTL	Capture/Compare/PWM 4
Fault0	17	I	TTL	PWM Fault 0
Fault1	41	I	TTL	PWM Fault 1
Fault2	16	I	TTL	PWM Fault 2
Fault3	84	I	TTL	PWM Fault 3
GND	9	-	Power	Ground reference for logic and I/O pins.
GND	15	-	Power	Ground reference for logic and I/O pins.
GND	21	-	Power	Ground reference for logic and I/O pins.
GND	33	-	Power	Ground reference for logic and I/O pins.
GND	39	-	Power	Ground reference for logic and I/O pins.
GND	45	-	Power	Ground reference for logic and I/O pins.
GND	54	-	Power	Ground reference for logic and I/O pins.
GND	57	-	Power	Ground reference for logic and I/O pins.
GND	63	-	Power	Ground reference for logic and I/O pins.
GND	69	-	Power	Ground reference for logic and I/O pins.
GND	82	-	Power	Ground reference for logic and I/O pins.
GND	87	-	Power	Ground reference for logic and I/O pins.
GND	94	-	Power	Ground reference for logic and I/O pins.

Pin Name	Pin Number	Pin Type	Buffer Type	Description
GNDA	4	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
$\overline{\text{HIB}}$	51	O	OD	An output that indicates the processor is in hibernate mode.
I2C0SCL	72	I/O	OD	I2C module 0 clock
I2C0SDA	65	I/O	OD	I2C module 0 data
I2C1SCL	34	I/O	OD	I2C module 1 clock
I2C1SDA	35	I/O	OD	I2C module 1 data
IDX0	40	I	TTL	QEI module 0 index
LDO	7	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. The LDO pin must also be connected to the VDD25 pins at the board level in addition to the decoupling capacitor(s).
NMI	89	I	TTL	Non maskable interrupt
OSC0	48	I	Analog	Main oscillator crystal input or an external clock reference input.
OSC1	49	O	Analog	Main oscillator crystal output.
PA0	26	I/O	TTL	GPIO port A bit 0
PA1	27	I/O	TTL	GPIO port A bit 1
PA2	28	I/O	TTL	GPIO port A bit 2
PA3	29	I/O	TTL	GPIO port A bit 3
PA4	30	I/O	TTL	GPIO port A bit 4
PA5	31	I/O	TTL	GPIO port A bit 5
PA6	34	I/O	TTL	GPIO port A bit 6
PA7	35	I/O	TTL	GPIO port A bit 7
PB0	66	I/O	TTL	GPIO port B bit 0
PB1	67	I/O	TTL	GPIO port B bit 1
PB2	72	I/O	TTL	GPIO port B bit 2
PB3	65	I/O	TTL	GPIO port B bit 3
PB4	92	I/O	TTL	GPIO port B bit 4
PB5	91	I/O	TTL	GPIO port B bit 5
PB6	90	I/O	TTL	GPIO port B bit 6
PB7	89	I/O	TTL	GPIO port B bit 7
PC0	80	I/O	TTL	GPIO port C bit 0
PC1	79	I/O	TTL	GPIO port C bit 1
PC2	78	I/O	TTL	GPIO port C bit 2
PC3	77	I/O	TTL	GPIO port C bit 3
PC4	25	I/O	TTL	GPIO port C bit 4
PC5	24	I/O	TTL	GPIO port C bit 5
PC6	23	I/O	TTL	GPIO port C bit 6
PC7	22	I/O	TTL	GPIO port C bit 7

Pin Name	Pin Number	Pin Type	Buffer Type	Description
PD0	10	I/O	TTL	GPIO port D bit 0
PD1	11	I/O	TTL	GPIO port D bit 1
PD2	12	I/O	TTL	GPIO port D bit 2
PD3	13	I/O	TTL	GPIO port D bit 3
PD4	97	I/O	Analog	GPIO port D bit 4
PD5	98	I/O	Analog	GPIO port D bit 5
PD6	99	I/O	Analog	GPIO port D bit 6
PD7	100	I/O	Analog	GPIO port D bit 7
PE0	74	I/O	TTL	GPIO port E bit 0
PE1	75	I/O	TTL	GPIO port E bit 1
PE2	95	I/O	TTL	GPIO port E bit 2
PE3	96	I/O	TTL	GPIO port E bit 3
PE4	6	I/O	Analog	GPIO port E bit 4
PE5	5	I/O	Analog	GPIO port E bit 5
PE6	2	I/O	Analog	GPIO port E bit 6
PE7	1	I/O	Analog	GPIO port E bit 7
PF0	47	I/O	TTL	GPIO port F bit 0
PF1	61	I/O	TTL	GPIO port F bit 1
PF2	60	I/O	TTL	GPIO port F bit 2
PF3	59	I/O	TTL	GPIO port F bit 3
PF4	58	I/O	TTL	GPIO port F bit 4
PF5	46	I/O	TTL	GPIO port F bit 5
PF6	43	I/O	TTL	GPIO port F bit 6
PF7	42	I/O	TTL	GPIO port F bit 7
PG0	19	I/O	TTL	GPIO port G bit 0
PG1	18	I/O	TTL	GPIO port G bit 1
PG2	17	I/O	TTL	GPIO port G bit 2
PG3	16	I/O	TTL	GPIO port G bit 3
PG4	41	I/O	TTL	GPIO port G bit 4
PG5	40	I/O	TTL	GPIO port G bit 5
PG6	37	I/O	TTL	GPIO port G bit 6
PG7	36	I/O	TTL	GPIO port G bit 7
PH0	86	I/O	TTL	GPIO port H bit 0
PH1	85	I/O	TTL	GPIO port H bit 1
PH2	84	I/O	TTL	GPIO port H bit 2
PH3	83	I/O	TTL	GPIO port H bit 3
PH4	76	I/O	TTL	GPIO port H bit 4
PhA0	43	I	TTL	QEI module 0 Phase A
PhB0	42	I	TTL	QEI module 0 Phase B
PWM0	19	O	TTL	PWM 0
PWM1	18	O	TTL	PWM 1
PWM2	86	O	TTL	PWM 2

Pin Name	Pin Number	Pin Type	Buffer Type	Description
PWM3	85	O	TTL	PWM 3
PWM4	60	O	TTL	PWM 4
PWM5	59	O	TTL	PWM 5
PWM6	37	O	TTL	PWM 6
PWM7	36	O	TTL	PWM 7
RST	64	I/O	TTL	System reset input.
SSI0Clk	28	I/O	TTL	SSI module 0 clock
SSI0Fss	29	I/O	TTL	SSI module 0 frame
SSI0Rx	30	I	TTL	SSI module 0 receive
SSI0Tx	31	O	TTL	SSI module 0 transmit
SSI1Clk	74	I/O	TTL	SSI module 1 clock
SSI1Fss	75	I/O	TTL	SSI module 1 frame
SSI1Rx	95	I	TTL	SSI module 1 receive
SSI1Tx	96	O	TTL	SSI module 1 transmit
SWCLK	80	I	TTL	JTAG/SWD CLK
SWDIO	79	I/O	TTL	JTAG TMS and SWDIO
SWO	77	O	TTL	JTAG TDO and SWO
TCK	80	I	TTL	JTAG/SWD CLK
TDI	78	I	TTL	JTAG TDI
TDO	77	O	TTL	JTAG TDO and SWO
TMS	79	I/O	TTL	JTAG TMS and SWDIO
U0Rx	26	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
U0Tx	27	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
U1Rx	12	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
U1Tx	13	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
USB0DM	70	I/O	Analog	Bidirectional differential data pin (D- per USB specification).
USB0DP	71	I/O	Analog	Bidirectional differential data pin (D+ per USB specification).
USB0EPEN	83	O	TTL	Used in Host mode to control an external power source to supply power to the USB bus.
USB0PFLT	76	I	TTL	Used in Host mode by an external power source to indicate an error state by that power source.
USB0RBIAS	73	I	Analog	9.1 KOhm resistor (1% precision) used internally for USB analog circuitry.
VBAT	55	-	Power	Power source for the Hibernation Module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation Module power-source supply.
VDD	8	-	Power	Positive supply for I/O and some logic.
VDD	20	-	Power	Positive supply for I/O and some logic.

Pin Name	Pin Number	Pin Type	Buffer Type	Description
VDD	32	-	Power	Positive supply for I/O and some logic.
VDD	44	-	Power	Positive supply for I/O and some logic.
VDD	56	-	Power	Positive supply for I/O and some logic.
VDD	68	-	Power	Positive supply for I/O and some logic.
VDD	81	-	Power	Positive supply for I/O and some logic.
VDD	93	-	Power	Positive supply for I/O and some logic.
VDD25	14	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDD25	38	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDD25	62	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDD25	88	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
VDDA	3	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
WAKE	50	I	-	An external input that brings the processor out of hibernate mode when asserted.
XOSC0	52	I	Analog	Hibernation Module oscillator crystal input or an external clock reference input. Note that this is either a 4.19-MHz crystal or a 32.768-kHz oscillator for the Hibernation Module RTC. See the CLKSEL bit in the **HIBCTL** register.
XOSC1	53	O	Analog	Hibernation Module oscillator crystal output.

Table 23-3. Signals by Function, Except for GPIO

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
ADC	ADC0	1	I	Analog	ADC 0 input
	ADC1	2	I	Analog	ADC 1 input
	ADC2	5	I	Analog	ADC 2 input
	ADC3	6	I	Analog	ADC 3 input
	ADC4	100	I	Analog	ADC 4 input
	ADC5	99	I	Analog	ADC 5 input
	ADC6	98	I	Analog	ADC 6 input
	ADC7	97	I	Analog	ADC 7 input
Analog Comparators	C0+	90	I	Analog	Analog comparator 0 positive input
	C0-	92	I	Analog	Analog comparator 0 negative input
	C0o	58	O	TTL	Analog comparator 0 output
	C1+	24	I	Analog	Analog comparator positive input

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
	C1-	91	I	Analog	Analog comparator 1 negative input
	C1o	46	O	TTL	Analog comparator 1 output
Controller Area Network	CAN0Rx	10	I	TTL	CAN module 0 receive
	CAN0Tx	11	O	TTL	CAN module 0 transmit
	CAN1Rx	47	I	TTL	CAN module 1 receive
	CAN1Tx	61	O	TTL	CAN module 1 transmit
General-Purpose Timers	CCP0	66	I/O	TTL	Capture/Compare/PWM 0
	CCP1	67	I/O	TTL	Capture/Compare/PWM 1
	CCP2	25	I/O	TTL	Capture/Compare/PWM 2
	CCP3	23	I/O	TTL	Capture/Compare/PWM 3
	CCP4	22	I/O	TTL	Capture/Compare/PWM 4
I2C	I2C0SCL	72	I/O	OD	I2C module 0 clock
	I2C0SDA	65	I/O	OD	I2C module 0 data
	I2C1SCL	34	I/O	OD	I2C module 1 clock
	I2C1SDA	35	I/O	OD	I2C module 1 data
JTAG/SWD/SWO	SWCLK	80	I	TTL	JTAG/SWD CLK
	SWDIO	79	I/O	TTL	JTAG TMS and SWDIO
	SWO	77	O	TTL	JTAG TDO and SWO
	TCK	80	I	TTL	JTAG/SWD CLK
	TDI	78	I	TTL	JTAG TDI
	TDO	77	O	TTL	JTAG TDO and SWO
	TMS	79	I/O	TTL	JTAG TMS and SWDIO
PWM	Fault0	17	I	TTL	PWM Fault 0
	Fault1	41	I	TTL	PWM Fault 1
	Fault2	16	I	TTL	PWM Fault 2
	Fault3	84	I	TTL	PWM Fault 3
	PWM0	19	O	TTL	PWM 0
	PWM1	18	O	TTL	PWM 1
	PWM2	86	O	TTL	PWM 2
	PWM3	85	O	TTL	PWM 3
	PWM4	60	O	TTL	PWM 4
	PWM5	59	O	TTL	PWM 5
	PWM6	37	O	TTL	PWM 6
	PWM7	36	O	TTL	PWM 7
Power	GND	9	-	Power	Ground reference for logic and I/O pins.
	GND	15	-	Power	Ground reference for logic and I/O pins.
	GND	21	-	Power	Ground reference for logic and I/O pins.
	GND	33	-	Power	Ground reference for logic and I/O pins.
	GND	39	-	Power	Ground reference for logic and I/O pins.
	GND	45	-	Power	Ground reference for logic and I/O pins.
	GND	54	-	Power	Ground reference for logic and I/O pins.
	GND	57	-	Power	Ground reference for logic and I/O pins.

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
	GND	63	-	Power	Ground reference for logic and I/O pins.
	GND	69	-	Power	Ground reference for logic and I/O pins.
	GND	82	-	Power	Ground reference for logic and I/O pins.
	GND	87	-	Power	Ground reference for logic and I/O pins.
	GND	94	-	Power	Ground reference for logic and I/O pins.
	GNDA	4	-	Power	The ground reference for the analog circuits (ADC, Analog Comparators, etc.). These are separated from GND to minimize the electrical noise contained on VDD from affecting the analog functions.
	HIB	51	O	OD	An output that indicates the processor is in hibernate mode.
	LDO	7	-	Power	Low drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater. The LDO pin must also be connected to the VDD25 pins at the board level in addition to the decoupling capacitor(s).
	VBAT	55	-	Power	Power source for the Hibernation Module. It is normally connected to the positive terminal of a battery and serves as the battery backup/Hibernation Module power-source supply.
	VDD	8	-	Power	Positive supply for I/O and some logic.
	VDD	20	-	Power	Positive supply for I/O and some logic.
	VDD	32	-	Power	Positive supply for I/O and some logic.
	VDD	44	-	Power	Positive supply for I/O and some logic.
	VDD	56	-	Power	Positive supply for I/O and some logic.
	VDD	68	-	Power	Positive supply for I/O and some logic.
	VDD	81	-	Power	Positive supply for I/O and some logic.
	VDD	93	-	Power	Positive supply for I/O and some logic.
	VDD25	14	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
	VDD25	38	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
	VDD25	62	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
	VDD25	88	-	Power	Positive supply for most of the logic function, including the processor core and most peripherals.
	VDDA	3	-	Power	The positive supply (3.3 V) for the analog circuits (ADC, Analog Comparators, etc.). These are separated from VDD to minimize the electrical noise contained on VDD from affecting the analog functions.
	WAKE	50	I	-	An external input that brings the processor out of hibernate mode when asserted.
QEI	IDX0	40	I	TTL	QEI module 0 index
	PhA0	43	I	TTL	QEI module 0 Phase A
	PhB0	42	I	TTL	QEI module 0 Phase B
SSI	SSI0Clk	28	I/O	TTL	SSI module 0 clock
	SSI0Fss	29	I/O	TTL	SSI module 0 frame

Function	Pin Name	Pin Number	Pin Type	Buffer Type	Description
	SSI0Rx	30	I	TTL	SSI module 0 receive
	SSI0Tx	31	O	TTL	SSI module 0 transmit
	SSI1Clk	74	I/O	TTL	SSI module 1 clock
	SSI1Fss	75	I/O	TTL	SSI module 1 frame
	SSI1Rx	95	I	TTL	SSI module 1 receive
	SSI1Tx	96	O	TTL	SSI module 1 transmit
System Control & Clocks	NMI	89	I	TTL	Non maskable interrupt
	OSC0	48	I	Analog	Main oscillator crystal input or an external clock reference input.
	OSC1	49	O	Analog	Main oscillator crystal output.
	RST	64	I/O	TTL	System reset input.
	USB0DM	70	I/O	Analog	Bidirectional differential data pin (D- per USB specification).
	USB0DP	71	I/O	Analog	Bidirectional differential data pin (D+ per USB specification).
	USB0RBIAS	73	I	Analog	9.1 KOhm resistor (1% precision) used internally for USB analog circuitry.
	XOSC0	52	I	Analog	Hibernation Module oscillator crystal input or an external clock reference input. Note that this is either a 4.19-MHz crystal or a 32.768-kHz oscillator for the Hibernation Module RTC. See the CLKSEL bit in the **HIBCTL** register.
	XOSC1	53	O	Analog	Hibernation Module oscillator crystal output.
UART	U0Rx	26	I	TTL	UART module 0 receive. When in IrDA mode, this signal has IrDA modulation.
	U0Tx	27	O	TTL	UART module 0 transmit. When in IrDA mode, this signal has IrDA modulation.
	U1Rx	12	I	TTL	UART module 1 receive. When in IrDA mode, this signal has IrDA modulation.
	U1Tx	13	O	TTL	UART module 1 transmit. When in IrDA mode, this signal has IrDA modulation.
USB	USB0EPEN	83	O	TTL	Used in Host mode to control an external power source to supply power to the USB bus.
	USB0PFLT	76	I	TTL	Used in Host mode by an external power source to indicate an error state by that power source.

Table 23-4. GPIO Pins and Alternate Functions

GPIO Pin	Pin Number	Multiplexed Function	Multiplexed Function
PA0	26	U0Rx	
PA1	27	U0Tx	
PA2	28	SSI0Clk	
PA3	29	SSI0Fss	
PA4	30	SSI0Rx	
PA5	31	SSI0Tx	
PA6	34	I2C1SCL	
PA7	35	I2C1SDA	

GPIO Pin	Pin Number	Multiplexed Function	Multiplexed Function
PB0	66	CCP0	
PB1	67	CCP1	
PB2	72	I2C0SCL	
PB3	65	I2C0SDA	
PB4	92	C0-	
PB5	91	C1-	
PB6	90	C0+	
PB7	89	NMI	
PC0	80	TCK	SWCLK
PC1	79	TMS	SWDIO
PC2	78	TDI	
PC3	77	TDO	SWO
PC4	25	CCP2	
PC5	24	C1+	
PC6	23	CCP3	
PC7	22	CCP4	
PD0	10	CAN0Rx	
PD1	11	CAN0Tx	
PD2	12	U1Rx	
PD3	13	U1Tx	
PD4	97	ADC7	
PD5	98	ADC6	
PD6	99	ADC5	
PD7	100	ADC4	
PE0	74	SSI1Clk	
PE1	75	SSI1Fss	
PE2	95	SSI1Rx	
PE3	96	SSI1Tx	
PE4	6	ADC3	
PE5	5	ADC2	
PE6	2	ADC1	
PE7	1	ADC0	
PF0	47	CAN1Rx	
PF1	61	CAN1Tx	
PF2	60	PWM4	
PF3	59	PWM5	
PF4	58	C0o	
PF5	46	C1o	
PF6	43	PhA0	
PF7	42	PhB0	
PG0	19	PWM0	
PG1	18	PWM1	

GPIO Pin	Pin Number	Multiplexed Function	Multiplexed Function
PG2	17	Fault0	
PG3	16	Fault2	
PG4	41	Fault1	
PG5	40	IDX0	
PG6	37	PWM6	
PG7	36	PWM7	
PH0	86	PWM2	
PH1	85	PWM3	
PH2	84	Fault3	
PH3	83	USB0EPEN	
PH4	76	USB0PFLT	

24 Operating Characteristics

Table 24-1. Temperature Characteristics

Characteristic[a]	Symbol	Value	Unit
Industrial operating temperature range	T_A	-40 to +85	°C

a. Maximum storage temperature is 150°C.

Table 24-2. Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance (junction to ambient)[a]	Θ_{JA}	32	°C/W
Average junction temperature[b]	T_J	$T_A + (P_{AVG} \bullet \Theta_{JA})$	°C

a. Junction to ambient thermal resistance θ_{JA} numbers are determined by a package simulator.

b. Power dissipation is a function of temperature.

25 Electrical Characteristics

25.1 DC Characteristics

25.1.1 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 25-1. Maximum Ratings

Characteristic [a]	Symbol	Value		Unit
		Min	Max	
I/O supply voltage (V_{DD})	V_{DD}	0	4	V
Core supply voltage (V_{DD25})	V_{DD25}	0	3	V
Analog supply voltage (V_{DDA})	V_{DDA}	0	4	V
Battery supply voltage (V_{BAT})	V_{BAT}	0	4	V
Input voltage	V_{IN}	-0.3	5.5	V
Maximum current per output pins	I	-	25	mA

a. Voltages are measured with respect to GND.

Important: This device contains circuitry to protect the inputs against damage due to high-static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either GND or V_{DD}).

25.1.2 Recommended DC Operating Conditions

For special high-current applications, the GPIO output buffers may be used with the following restrictions. With the GPIO pins configured as 8-mA output drivers, a total of four GPIO outputs may be used to sink current loads up to 18 mA each. At 18-mA sink current loading, the V_{OL} value is specified as 1.2 V. The high-current GPIO package pins must be selected such that there are only a maximum of two per side of the physical package with the total number of high-current GPIO outputs not exceeding four for the entire package.

Table 25-2. Recommended DC Operating Conditions

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{DD}	I/O supply voltage	3.0	3.3	3.6	V
V_{DD25}	Core supply voltage	2.25	2.5	2.75	V
V_{DDA}	Analog supply voltage	3.0	3.3	3.6	V
V_{BAT}	Battery supply voltage	2.3	3.0	3.6	V
V_{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

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{OH}[a]	High-level output voltage	2.4	-	-	V
V_{OL}[a]	Low-level output voltage	-	-	0.4	V
I_{OH}	High-level source current, V_{OH}=2.4 V				
	2-mA Drive	2.0	-	-	mA
	4-mA Drive	4.0	-	-	mA
	8-mA Drive	8.0	-	-	mA
I_{OL}	Low-level sink current, V_{OL}=0.4 V				
	2-mA Drive	2.0	-	-	mA
	4-mA Drive	4.0	-	-	mA
	8-mA Drive	8.0	-	-	mA

a. V_{OL} and V_{OH} shift to 1.2 V when using high-current GPIOs.

25.1.3 On-Chip Low Drop-Out (LDO) Regulator Characteristics

Table 25-3. LDO Regulator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{LDOOUT}	Programmable internal (logic) power supply output value	2.25	2.5	2.75	V
	Output voltage accuracy	-	2%	-	%
t_{PON}	Power-on time	-	-	100	µs
t_{ON}	Time on	-	-	200	µs
t_{OFF}	Time off	-	-	100	µs
V_{STEP}	Step programming incremental voltage	-	50	-	mV
C_{LDO}	External filter capacitor size for internal power supply	1.0	-	3.0	µF

25.1.4 Power Specifications

The power measurements specified in the tables that follow are run on the core processor using SRAM with the following specifications (except as noted):

- V_{DD} = 3.3 V
- V_{DD25} = 2.50 V
- V_{BAT} = 3.0 V
- V_{DDA} = 3.3 V
- Temperature = 25°C
- Clock Source (MOSC) =3.579545 MHz Crystal Oscillator
- Main oscillator (MOSC) = enabled
- Internal oscillator (IOSC) = disabled

Table 25-4. Detailed Power Specifications

Parameter	Parameter Name	Conditions	3.3 V V_{DD}, V_{DDA}		2.5 V V_{DD25}		3.0 V V_{BAT}		Unit
			Nom	Max	Nom	Max	Nom	Max	
I_{DD_RUN}	Run mode 1 (Flash loop)	V_{DD25} = 2.50 V Code= while(1){} executed in Flash Peripherals = All ON System Clock = 50 MHz (with PLL)	9.5	pending[a]	108	pending[a]	0	pending[a]	mA
	Run mode 2 (Flash loop)	V_{DD25} = 2.50 V Code= while(1){} executed in Flash Peripherals = All OFF System Clock = 50 MHz (with PLL)	<0.001	pending[a]	53	pending[a]	0	pending[a]	mA
	Run mode 1 (SRAM loop)	V_{DD25} = 2.50 V Code= while(1){} executed in SRAM Peripherals = All ON System Clock = 50 MHz (with PLL)	9.5	pending[a]	102	pending[a]	0	pending[a]	mA
	Run mode 2 (SRAM loop)	V_{DD25} = 2.50 V Code= while(1){} executed in SRAM Peripherals = All OFF System Clock = 50 MHz (with PLL)	<0.001	pending[a]	47	pending[a]	0	pending[a]	mA
I_{DD_SLEEP}	Sleep mode	V_{DD25} = 2.50 V Peripherals = All OFF System Clock = 50 MHz (with PLL)	<0.001	pending[a]	17	pending[a]	0	pending[a]	mA
$I_{DD\_DEEPSLEEP}$	Deep-Sleep mode	LDO = 2.25 V Peripherals = All OFF System Clock = IOSC30KHZ/64	0.14	pending[a]	0.18	pending[a]	0	pending[a]	mA
$I_{DD\_HIBERNATE}$	Hibernate mode	V_{BAT} = 3.0 V V_{DD} = 0 V V_{DD25} = 0 V V_{DDA} = 0 V Peripherals = All OFF System Clock = OFF Hibernate Module = 32 kHz	0	0	0	0	16	pending[a]	µA

a. Pending characterization completion.

25.1.5 Flash Memory Characteristics

Table 25-5. Flash Memory Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
PE_{CYC}	Number of guaranteed program/erase cycles before failure[a]	10,000	100,000	-	cycles
T_{RET}	Data retention at average operating temperature of 85°C	10	-	-	years
T_{PROG}	Word program time	20	-	-	µs
T_{ERASE}	Page erase time	20	-	-	ms
T_{ME}	Mass erase time	200	-	-	ms

a. A program/erase cycle is defined as switching the bits from 1-> 0 -> 1.

25.1.6 Hibernation

Table 25-6. Hibernation Module DC Characteristics

Parameter	Parameter Name	Value	Unit
V_{LOWBAT}	Low battery detect voltage	2.35	V
R_{WAKEPU}	$\overline{WAKE}$ internal pull-up resistor	200	kΩ

25.1.7 USB

The Stellaris® USB controller DC electrical specifications are compliant with the “Universal Serial Bus Specification Rev. 2.0” (full-speed and low-speed support). Some components of the USB system are integrated within the LM3S5749 microcontroller and specific to the Stellaris® microcontroller design. These components are specified in Table 25-7 on page 752.

Table 25-7. USB Controller DC Characteristics

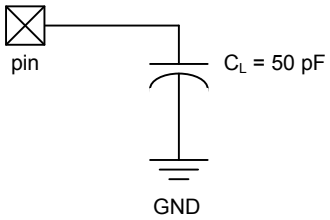
Parameter	Parameter Name	Value	Unit
R_{BIAS}	Value of the pull-down resistor on the USB0RBIAS pin	9.1K ± 1 %	Ω

25.2 AC Characteristics

25.2.1 Load Conditions

Unless otherwise specified, the following conditions are true for all timing measurements. Timing measurements are for 4-mA drive strength.

Figure 25-1. Load Conditions



25.2.2 Clocks

Table 25-8. Phase Locked Loop (PLL) Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$f_{ref\_crystal}$	Crystal reference[a]	3.579545	-	16.384	MHz
f_{ref_ext}	External clock reference[a]	3.579545	-	16.384	MHz
f_{pll}	PLL frequency[b]	-	400	-	MHz
T_{READY}	PLL lock time	-	-	0.5	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 25-9. Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
f_{IOSC}	Internal 12 MHz oscillator frequency	8.4	12	15.6	MHz
$f_{IOSC30KHZ}$	Internal 30 KHz oscillator frequency	21	30	39	KHz
f_{XOSC}	Hibernation module oscillator frequency	-	4.194304	-	MHz
f_{XOSC_XTAL}	Crystal reference for hibernation oscillator	-	4.194304	-	MHz
f_{XOSC_EXT}	External clock reference for hibernation module	-	32.768	-	KHz
f_{MOSC}	Main oscillator frequency	1	-	16.384	MHz
t_{MOSC_per}	Main oscillator period	61	-	1000	ns
$f_{ref\_crystal\_bypass}$	Crystal reference using the main oscillator (PLL in BYPASS mode)[a]	1	-	16.384	MHz
$f_{ref\_ext\_bypass}$	External clock reference (PLL in BYPASS mode)[a]	0	-	50	MHz
f_{system_clock}	System clock	0	-	50	MHz

a. The ADC must be clocked from the PLL or directly from a 14-MHz to 18-MHz clock source to operate properly.

Table 25-10. Crystal Characteristics

Parameter Name	Value						Units
Frequency	16	12	8	6	4	3.5	MHz
Frequency tolerance	±50	±50	±50	±50	±50	±50	ppm
Aging	±5	±5	±5	±5	±5	±5	ppm/yr
Oscillation mode	Parallel	Parallel	Parallel	Parallel	Parallel	Parallel	-
Temperature stability (-40°C to 85°C)	±25	±25	±25	±25	±25	±25	ppm
Motional capacitance (typ)	13.9	18.5	27.8	37.0	55.6	63.5	pF
Motional inductance (typ)	7.15	9.5	14.3	19.1	28.6	32.7	mH
Equivalent series resistance (max)	80	100	120	160	200	220	Ω
Shunt capacitance (max)	10	10	10	10	10	10	pF
Load capacitance (typ)	16	16	16	16	16	16	pF
Drive level (typ)	100	100	100	100	100	100	µW

25.2.3 JTAG and Boundary Scan

Table 25-11. JTAG Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J1	f_{TCK}	TCK operational clock frequency	0	-	10	MHz
J2	t_{TCK}	TCK operational clock period	100	-	-	ns
J3	t_{TCK_LOW}	TCK clock Low time	-	t_{TCK}	-	ns
J4	t_{TCK_HIGH}	TCK clock High time	-	t_{TCK}	-	ns
J5	t_{TCK_R}	TCK rise time	0	-	10	ns
J6	t_{TCK_F}	TCK fall time	0	-	10	ns
J7	t_{TMS_SU}	TMS setup time to TCK rise	20	-	-	ns
J8	t_{TMS_HLD}	TMS hold time from TCK rise	20	-	-	ns
J9	t_{TDI_SU}	TDI setup time to TCK rise	25	-	-	ns
J10	t_{TDI_HLD}	TDI hold time from TCK rise	25	-	-	ns
J11 t_{TDO_ZDV}	TCK fall to Data Valid from High-Z	2-mA drive	-	23	35	ns
		4-mA drive		15	26	ns
		8-mA drive		14	25	ns
		8-mA drive with slew rate control		18	29	ns
J12 t_{TDO_DV}	TCK fall to Data Valid from Data Valid	2-mA drive	-	21	35	ns
		4-mA drive		14	25	ns
		8-mA drive		13	24	ns
		8-mA drive with slew rate control		18	28	ns
J13 t_{TDO_DVZ}	TCK fall to High-Z from Data Valid	2-mA drive	-	9	11	ns
		4-mA drive		7	9	ns
		8-mA drive		6	8	ns
		8-mA drive with slew rate control		7	9	ns

Figure 25-2. JTAG Test Clock Input Timing

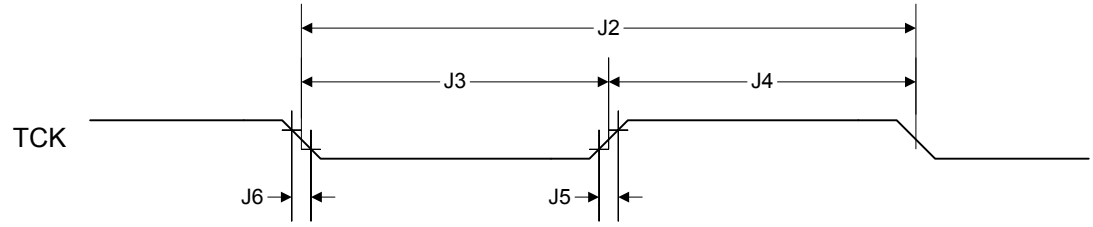
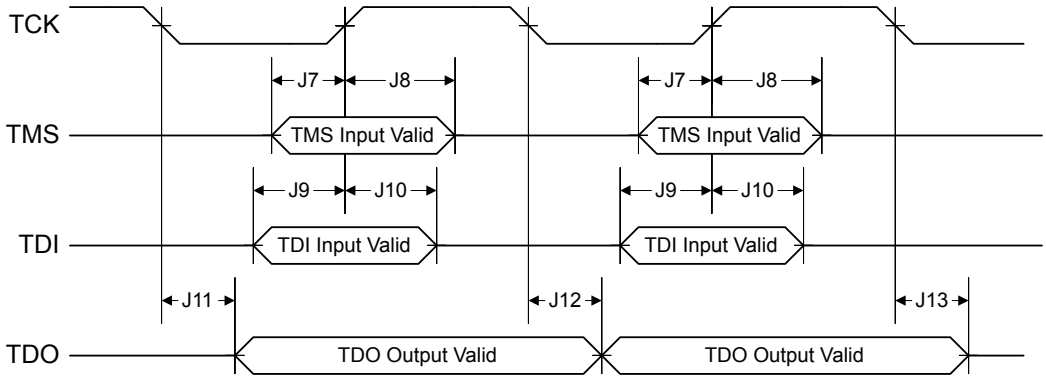


Figure 25-3. JTAG Test Access Port (TAP) Timing



25.2.4 Reset

Table 25-12. Reset Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
R1	V_{TH}	Reset threshold	-	2.0	-	V
R2	V_{BTH}	Brown-Out threshold	2.85	2.9	2.95	V
R3	T_{POR}	Power-On Reset timeout	-	10	-	ms
R4	T_{BOR}	Brown-Out timeout	-	500	-	µs
R5	T_{IRPOR}	Internal reset timeout after POR	6	-	11	ms
R6	T_{IRBOR}	Internal reset timeout after BOR[a]	0	-	1	µs
R7	T_{IRHWR}	Internal reset timeout after hardware reset ($\overline{RST}$ pin)	0	-	1	ms
R8	T_{IRSWR}	Internal reset timeout after software-initiated system reset [a]	2.5	-	20	µs
R9	T_{IRWDR}	Internal reset timeout after watchdog reset[a]	2.5	-	20	µs
R10	$T_{VDDRISE}$	Supply voltage (V_{DD}) rise time (0V-3.3V)	-	-	250	ms
R11	T_{MIN}	Minimum $\overline{RST}$ pulse width	2	-	-	µs

a. 20 * t_{MOSC_per}

Figure 25-4. External Reset Timing ($\overline{RST}$)

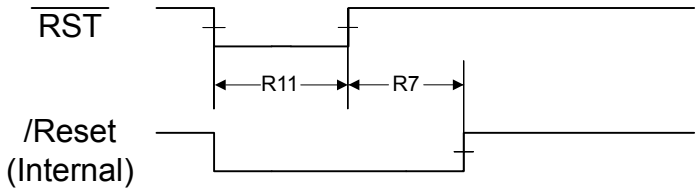


Figure 25-5. Power-On Reset Timing

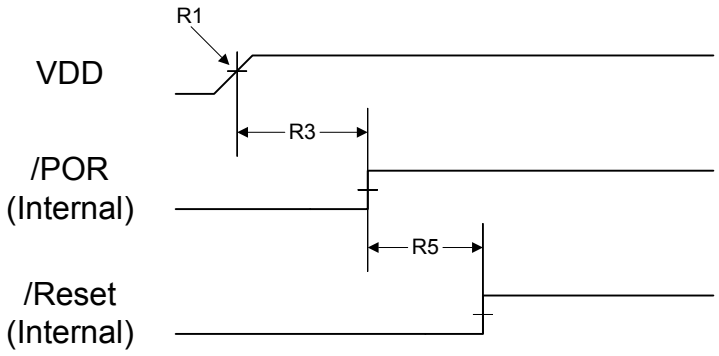


Figure 25-6. Brown-Out Reset Timing

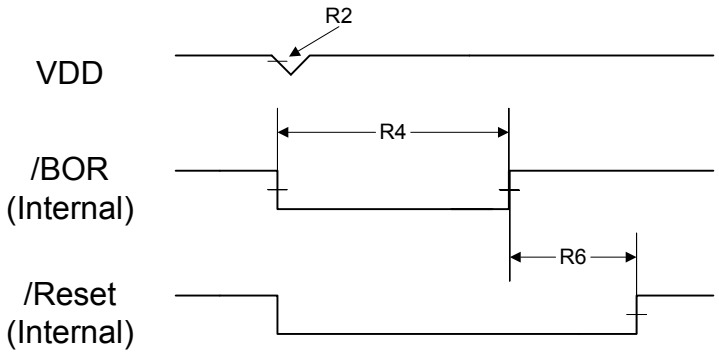


Figure 25-7. Software Reset Timing

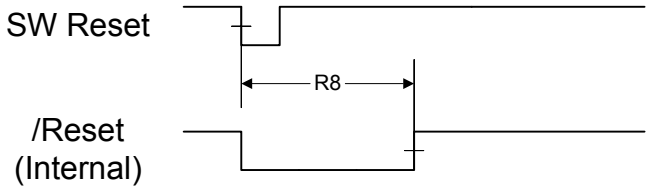
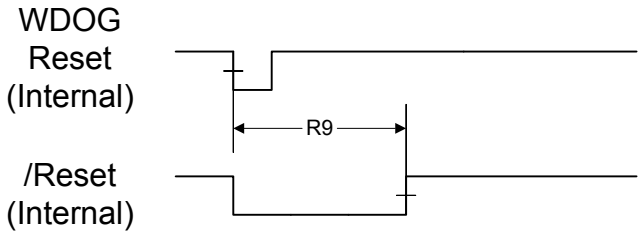


Figure 25-8. Watchdog Reset Timing



25.2.5 Hibernation Module

The Hibernation Module requires special system implementation considerations since it is intended to power-down all other sections of its host device. The system power-supply distribution and interfaces to the device must be driven to 0 V_{DC} or powered down with the same external voltage regulator controlled by $\overline{\text{HIB}}$.

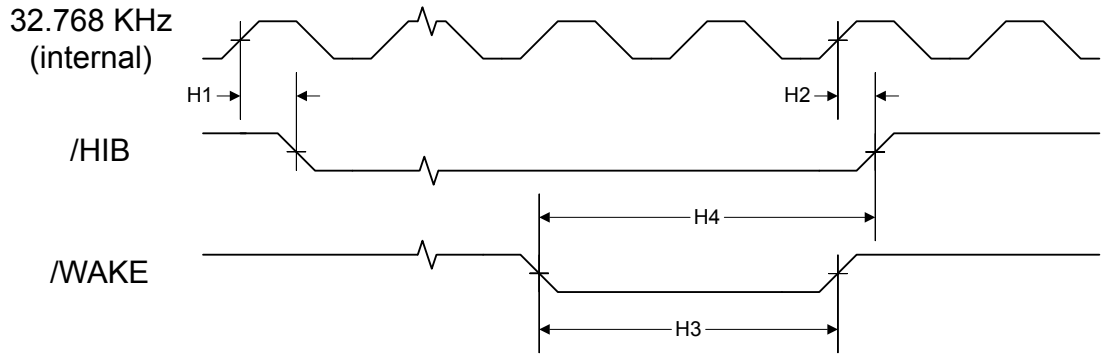
The external voltage regulators controlled by HIB must have a settling time of 250 µs or less.

Table 25-13. Hibernation Module AC Characteristics

Parameter No	Parameter	Parameter Name	Min	Nom	Max	Unit
H1	t_{HIB_LOW}	Internal 32.768 KHz clock reference rising edge to /HIB asserted	-	200	-	µs
H2	t_{HIB_HIGH}	Internal 32.768 KHz clock reference rising edge to /HIB deasserted	-	30	-	µs
H3	t_{WAKE_ASSERT}	/WAKE assertion time	62	-	-	µs
H4	$t_{WAKETOHIB}$	/WAKE assert to /HIB desassert	62	-	124	µs
H5	t_{XOSC_SETTLE}	XOSC settling time[a]	20	-	-	ms
H6	$t_{HIB\_REG\_WRITE}$	Time for a write to non-volatile registers in HIB module to complete	92	-	-	µs
H7	$t_{HIB\_TO\_VDD}$	HIB deassert to VDD and VDD25 at minimum operational level	-	-	250	µs

a. This parameter is highly sensitive to PCB layout and trace lengths, which may make this parameter time longer. Care must be taken in PCB design to minimize trace lengths and RLC (resistance, inductance, capacitance).

Figure 25-9. Hibernation Module Timing



25.2.6 General-Purpose I/O (GPIO)

Note: All GPIOs are 5 V-tolerant.

Table 25-14. 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
R_{GPIOPU}	GPIO internal pull-up resistor	Pull-up enabled	50	-	110	kΩ
R_{GPIOPD}	GPIO internal pull-down resistor	Pull-down enabled	55	-	180	kΩ

25.2.7 Analog-to-Digital Converter

Table 25-15. ADC Characteristics[a]

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{ADCIN}	Maximum single-ended, full-scale analog input voltage	-	-	3.0	V
	Minimum single-ended, full-scale analog input voltage	-	-	0	V
	Maximum differential, full-scale analog input voltage	-	-	1.5	V
	Minimum differential, full-scale analog input voltage	-	-	-1.5	V
C_{ADCIN}	Equivalent input capacitance	-	1	-	pF
N	Resolution	-	10	-	bits
f_{ADC}	ADC internal clock frequency	14	16	18	MHz
$t_{ADCCONV}$	Conversion time	-	-	16	t_{ADC}cycles[b]
$f_{ADCCONV}$	Conversion rate	875	1000	1125	k samples/s
INL	Integral nonlinearity	-	-	±1	LSB
DNL	Differential nonlinearity	-	-	±1	LSB
OFF	Offset	-	-	±1	LSB
GAIN	Gain	-	-	±1	LSB

a. The ADC reference voltage is 3.0 V. This reference voltage is internally generated from the 3.3 VDDA supply by a band gap circuit.

b. $t_{ADC} = 1/f_{ADC\ clock}$

25.2.8 Synchronous Serial Interface (SSI)

Table 25-16. SSI Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
S1	t_{clk_per}	SSIClk cycle time	2	-	65024	system clocks
S2	t_{clk_high}	SSIClk high time	-	0.5	-	t clk_per
S3	t_{clk_low}	SSIClk low time	-	0.5	-	t clk_per
S4	t_{clkrf}	SSIClk rise/fall time	-	7.4	26	ns
S5	t_{DMd}	Data from master valid delay time	0	-	20	ns
S6	t_{DMs}	Data from master setup time	20	-	-	ns
S7	t_{DMh}	Data from master hold time	40	-	-	ns
S8	t_{DSs}	Data from slave setup time	20	-	-	ns
S9	t_{DSh}	Data from slave hold time	40	-	-	ns

Figure 25-10. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement

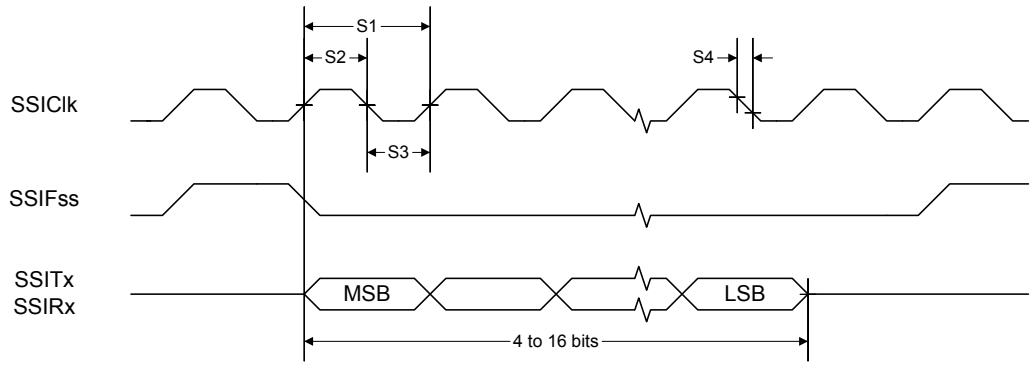


Figure 25-11. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer

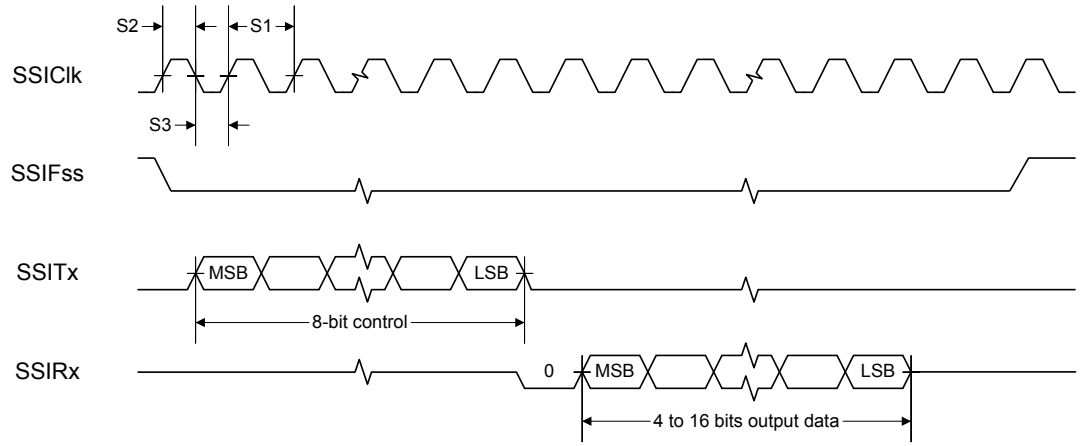
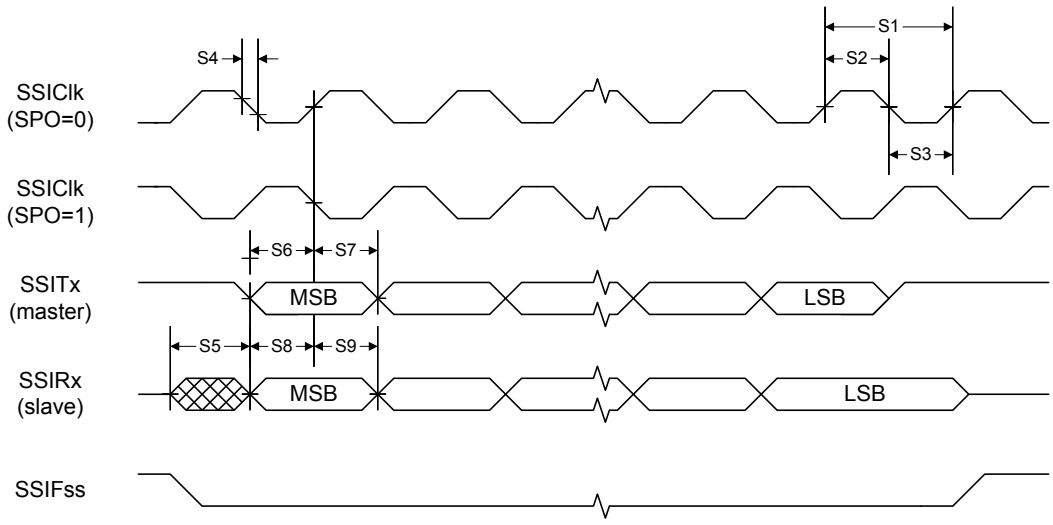


Figure 25-12. SSI Timing for SPI Frame Format (FRF=00), with SPH=1



25.2.9 Inter-Integrated Circuit (I^2C) Interface

Table 25-17. I^2C Characteristics

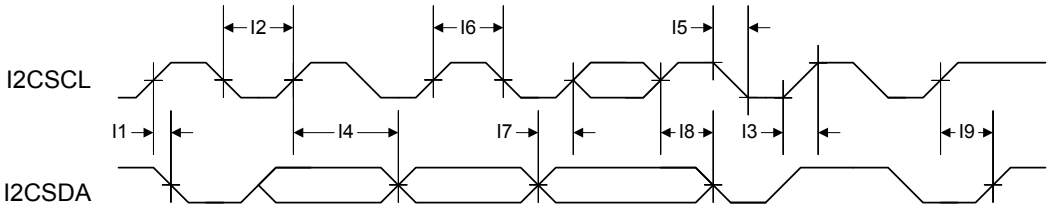
Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
I1[a]	t_{SCH}	Start condition hold time	36	-	-	system clocks
I2[a]	t_{LP}	Clock Low period	36	-	-	system clocks
I3[b]	t_{SRT}	I2CSCL/I2CSDA rise time (V_{IL} =0.5 V to V_{IH} =2.4 V)	-	-	(see note b)	ns
I4[a]	t_{DH}	Data hold time	2	-	-	system clocks
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 25-13. I²C Timing



25.2.10 Universal Serial Bus (USB) Controller

The Stellaris® USB controller AC electrical specifications are compliant with the “Universal Serial Bus Specification Rev. 2.0” (full-speed and low-speed support).

25.2.11 Analog Comparator

Table 25-18. Analog Comparator Characteristics

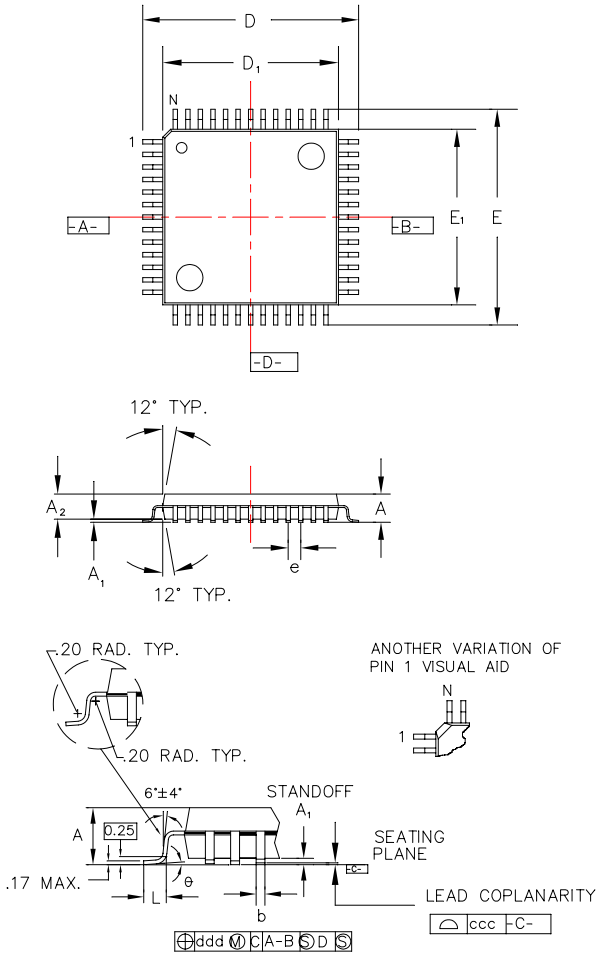
Parameter	Parameter Name	Min	Nom	Max	Unit
V_{OS}	Input offset voltage	-	±10	±25	mV
V_{CM}	Input common mode voltage range	0	-	V_{DD}-1.5	V
C_{MRR}	Common mode rejection ratio	50	-	-	dB
T_{RT}	Response time	-	-	1	µs
T_{MC}	Comparator mode change to Output Valid	-	-	10	µs

Table 25-19. 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

26 Package Information

Figure 26-1. 100-Pin LQFP Package



Note: The following notes apply to the package drawing.

1. All dimensions shown in mm.
2. Dimensions shown are nominal with tolerances indicated.
3. Foot length 'L' is measured at gage plane 0.25 mm above seating plane.

Body +2.00 mm Footprint, 1.4 mm package thickness		
Symbols	Leads	100L
A	Max.	1.60
A_1	-	0.05 Min./0.15 Max.
A_2	±0.05	1.40
D	±0.20	16.00
D_1	±0.05	14.00
E	±0.20	16.00
E_1	±0.05	14.00
L	+0.15/-0.10	0.60
e	Basic	0.50
b	+0.05	0.22
θ	-	0˚-7˚
ddd	Max.	0.08
ccc	Max.	0.08
JEDEC Reference Drawing		MS-026
Variation Designator		BED

A Boot Loader

A.1 Boot Loader

The Stellaris® Boot Loader is executed from the ROM when flash is empty and is used to download code to the flash memory of a device without the use of a debug interface. The boot loader uses a simple packet interface to provide synchronous communication with the device. The boot loader runs off the internal oscillator and does not enable the PLL, so its speed is determined by the speed of the internal oscillator. The following serial interfaces can be used:

- UART0
- SSI0
- I^2C0

For simplicity, both the data format and communication protocol are identical for all serial interfaces.

See the *Stellaris® Boot Loader User's Guide* for information on the boot loader software.

A.2 Interfaces

Once communication with the boot loader is established via one of the serial interfaces, that interface is used until the boot loader is reset or new code takes over. For example, once you start communicating using the SSI port, communications with the boot loader via the UART are disabled until the device is reset.

A.2.1 UART

The Universal Asynchronous Receivers/Transmitters (UART) communication uses a fixed serial format of 8 bits of data, no parity, and 1 stop bit. The baud rate used for communication is automatically detected by the boot loader and can be any valid baud rate supported by the host and the device. The auto detection sequence requires that the baud rate should be no more than 1/32 the internal oscillator frequency of the board that is running the boot loader (which is at least 8.4 MHz, providing support for up to 262,500 baud). This is actually the same as the hardware limitation for the maximum baud rate for any UART on a Stellaris® device which is calculated as follows:

```
Max Baud Rate = System Clock Frequency / 16
```

In order to determine the baud rate, the boot loader needs to determine the relationship between the internal oscillator and the baud rate. This is enough information for the boot loader to configure its UART to the same baud rate as the host. This automatic baud-rate detection allows the host to use any valid baud rate that it wants to communicate with the device.

The method used to perform this automatic synchronization relies on the host sending the boot loader two bytes that are both 0x55. This generates a series of pulses to the boot loader that it can use to calculate the ratios needed to program the UART to match the host's baud rate. After the host sends the pattern, it attempts to read back one byte of data from the UART. The boot loader returns the value of 0xCC to indicate successful detection of the baud rate. If this byte is not received after at least twice the time required to transfer the two bytes, the host can resend another pattern of 0x55, 0x55, and wait for the 0xCC byte again until the boot loader acknowledges that it has received a synchronization pattern correctly. For example, the time to wait for data back from the boot loader should be calculated as at least 2*(20(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 442 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 the internal oscillator frequency of the board running the boot loader (which is at least 8.4 MHz, providing support for up to 700 KHz).. Since the host device is the master, the SSI on the boot loader device does not need to determine the clock as it is provided directly by the host.

A.2.3 I^2C

The Inter-Integrated Circuit (I^2C) port operates in slave mode with a slave address of 0x42. The I^2C port will work at both 100 Khz and 400 KHz I^2C clock frequency. Since the host device is the master, the I^2C on the boot loader device does not need to determine the clock as it is provided directly by the host.

A.3 Packet Handling

All communications, with the exception of the UART auto-baud, are done via defined packets that are acknowledged (ACK) or not acknowledged (NAK) by the devices. The packets use the same format for receiving and sending packets, including the method used to acknowledge successful or unsuccessful reception of a packet.

A.3.1 Packet Format

All packets sent and received from the device use the following byte-packed format.

```
struct
{
 unsigned char ucSize;
 unsigned char ucCheckSum;
 unsigned char Data[];
};
```

ucSize	The first byte received holds the total size of the transfer including the size and checksum bytes.
ucChecksum	This holds a simple checksum of the bytes in the data buffer only. The algorithm is Data[0]+Data[1]+…+ Data[`ucSize`-3].
Data	This is the raw data intended for the device, which is formatted in some form of command interface. There should be `ucSize`–2 bytes of data provided in this buffer to or from the device.

A.3.2 Sending Packets

The actual bytes of the packet can be sent individually or all at once; the only limitation is that commands that cause flash memory access should limit the download sizes to prevent losing bytes during flash programming. This limitation is discussed further in the section that describes the boot loader command, COMMAND_SEND_DATA (see “COMMAND_SEND_DATA (0x24)” on page 767).

Once the packet has been formatted correctly by the host, it should be sent out over the UART or SSI interface. Then the host should poll the UART or SSI interface for the first non-zero data returned from the device. The first non-zero byte will either be an ACK (0xCC) or a NAK (0x33) byte from

the device indicating the packet was received successfully (ACK) or unsuccessfully (NAK). This does not indicate that the actual contents of the command issued in the data portion of the packet were valid, just that the packet was received correctly.

A.3.3 Receiving Packets

The boot loader sends a packet of data in the same format that it receives a packet. The boot loader may transfer leading zero data before the first actual byte of data is sent out. The first non-zero byte is the size of the packet followed by a checksum byte, and finally followed by the data itself. There is no break in the data after the first non-zero byte is sent from the boot loader. Once the device communicating with the boot loader receives all the bytes, it must either ACK or NAK the packet to indicate that the transmission was successful. The appropriate response after sending a NAK to the boot loader is to resend the command that failed and request the data again. If needed, the host may send leading zeros before sending down the ACK/NAK signal to the boot loader, as the boot loader only accepts the first non-zero data as a valid response. This zero padding is needed by the SSI interface in order to receive data to or from the boot loader.

A.4 Commands

The next section defines the list of commands that can be sent to the boot loader. The first byte of the data should always be one of the defined commands, followed by data or parameters as determined by the command that is sent.

A.4.1 COMMAND_PING (0X20)

This command simply accepts the command and sets the global status to success. The format of the packet is as follows:

```
Byte[0] = 0x03;
Byte[1] = checksum(Byte[2]);
Byte[2] = COMMAND_PING;
```

The ping command has 3 bytes and the value for `COMMAND_PING` is 0x20 and the checksum of one byte is that same byte, making Byte[1] also 0x20. Since the ping command has no real return status, the receipt of an ACK can be interpreted as a successful ping to the boot loader.

A.4.2 COMMAND_DOWNLOAD (0x21)

This command is sent to the boot loader to indicate where to store data and how many bytes will be sent by the `COMMAND_SEND_DATA` commands that follow. The command consists of two 32-bit values that are both transferred MSB first. The first 32-bit value is the address to start programming data into, while the second is the 32-bit size of the data that will be sent. This command also triggers an erase of the full area to be programmed so this command takes longer than other commands. This results in a longer time to receive the ACK/NAK back from the board. This command should be followed by a `COMMAND_GET_STATUS` to ensure that the Program Address and Program size are valid for the device running the boot loader.

The format of the packet to send this command is 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.4.3 COMMAND_RUN (0x22)

This command is used to tell the boot loader to execute from the address passed as the parameter in this command. This command consists of a single 32-bit value that is interpreted as the address to execute. The 32-bit value is transmitted MSB first and the boot loader responds with an ACK signal back to the host device before actually executing the code at the given address. This allows the host to know that the command was received successfully and the code is now running.

```
Byte[0] = 7
Byte[1] = checksum(Bytes[2:6])
Byte[2] = COMMAND_RUN
Byte[3] = Execute Address[31:24]
Byte[4] = Execute Address[23:16]
Byte[5] = Execute Address[15:8]
Byte[6] = Execute Address[7:0]
```

A.4.4 COMMAND_GET_STATUS (0x23)

This command returns the status of the last command that was issued. Typically, this command should be sent after every command to ensure that the previous command was successful or to properly respond to a failure. The command requires one byte in the data of the packet and should be followed by reading a packet with one byte of data that contains a status code. The last step is to ACK or NAK the received data so the boot loader knows that the data has been read.

```
Byte[0] = 0x03
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_GET_STATUS
```

A.4.5 COMMAND_SEND_DATA (0x24)

This command should only follow a COMMAND_DOWNLOAD command or another COMMAND_SEND_DATA command if more data is needed. Consecutive send data commands automatically increment address and continue programming from the previous location. For packets which do not contain the final portion of the downloaded data, a multiple of four bytes should always be transferred. The command terminates programming once the number of bytes indicated by the COMMAND_DOWNLOAD command has been received. Each time this function is called it should be followed by a COMMAND_GET_STATUS to ensure that the data was successfully programmed into the flash. If the boot loader sends a NAK to this command, the boot loader does not increment the current address to allow retransmission of the previous data. The following example shows a COMMAND_SEND_DATA packet with 8 bytes of packet data:

```
Byte[0] = 11
Byte[1] = checksum(Bytes[2:10])
Byte[2] = COMMAND_SEND_DATA
Byte[3] = Data[0]
Byte[4] = Data[1]
Byte[5] = Data[2]
Byte[6] = Data[3]
```

```
Byte[7] = Data[4]
Byte[8] = Data[5]
Byte[9] = Data[6]
Byte[10] = Data[7]
```

A.4.6 COMMAND_RESET (0x25)

This command is used to tell the boot loader device to reset. Unlike the COMMAND_RUN command, this allows the initial stack pointer to be read by the hardware and set up for the new code. It can also be used to reset the boot loader if a critical error occurs and the host device wants to restart communication with the boot loader.

```
Byte[0] = 3
Byte[1] = checksum(Byte[2])
Byte[2] = COMMAND_RESET
```

The boot loader responds with an ACK signal back to the host device before actually executing the software reset to the device running the boot loader. This allows the host to know that the command was received successfully and the part will be reset.

B ROM DriverLib Functions

B.1 DriverLib Functions Included in the Integrated ROM

The Stellaris® Peripheral Driver Library (DriverLib) APIs that are available in the integrated ROM of the Stellaris® family of devices are listed below. The detailed description of each function is available in the *Stellaris® ROM User's Guide*.

ROM_ADCHardwareOversampleConfigure
    // Configures the hardware oversampling factor of the ADC.

ROM_ADCIntClear
    // Clears sample sequence interrupt source.

ROM_ADCIntDisable
    // Disables a sample sequence interrupt.

ROM_ADCIntEnable
    // Enables a sample sequence interrupt.

ROM_ADCIntStatus
    // Gets the current interrupt status.

ROM_ADCProcessorTrigger
    // Causes a processor trigger for a sample sequence.

ROM_ADCSequenceConfigure
    // Configures the trigger source and priority of a sample sequence.

ROM_ADCSequenceDataGet
    // Gets the captured data for a sample sequence.

ROM_ADCSequenceDisable
    // Disables a sample sequence.

ROM_ADCSequenceEnable
    // Enables a sample sequence.

ROM_ADCSequenceOverflow
    // Determines if a sample sequence overflow occurred.

ROM_ADCSequenceOverflowClear
    // Clears the overflow condition on a sample sequence.

ROM_ADCSequenceStepConfigure
    // Configure a step of the sample sequencer.

ROM_ADCSequenceUnderflow
    // Determines if a sample sequence underflow occurred.

ROM_ADCSequenceUnderflowClear
    // Clears the underflow condition on a sample sequence.

ROM_ComparatorConfigure
    // Configures a comparator.

ROM_ComparatorIntClear
    // Clears a comparator interrupt.

ROM_ComparatorIntDisable
    // Disables the comparator interrupt.

ROM_ComparatorIntEnable
    // Enables the comparator interrupt.

ROM_ComparatorIntStatus
    // Gets the current interrupt status.

ROM_ComparatorRefSet
    // Sets the internal reference voltage.

ROM_ComparatorValueGet
    // Gets the current comparator output value.

ROM_FlashErase
    // Erases a block of flash.

ROM_FlashIntClear
    // Clears flash controller interrupt sources.

ROM_FlashIntDisable
    // Disables individual flash controller interrupt sources.

ROM_FlashIntEnable
    // Enables individual flash controller interrupt sources.

ROM_FlashIntGetStatus
    // Gets the current interrupt status.

ROM_FlashProgram
    // Programs flash.

ROM_FlashProtectGet
    // Gets the protection setting for a block of flash.

ROM_FlashProtectSave
    // Saves the flash protection settings.

ROM_FlashProtectSet
    // Sets the protection setting for a block of flash.

ROM_FlashUsecGet
    // Gets the number of processor clocks per micro-second.

ROM_FlashUsecSet
    // Sets the number of processor clocks per micro-second.

ROM_FlashUserGet
    // Gets the user registers.

ROM_FlashUserSave
    // Saves the user registers.

ROM_FlashUserSet
    // Sets the user registers.

ROM_GPIODirModeGet
    // Gets the direction and mode of a pin.

ROM_GPIODirModeSet
    // Sets the direction and mode of the specified pin(s).

ROM_GPIOIntTypeGet
    // Gets the interrupt type for a pin.

ROM_GPIOIntTypeSet
    // Sets the interrupt type for the specified pin(s).

ROM_GPIOPadConfigGet
    // Gets the pad configuration for a pin.

ROM_GPIOPadConfigSet
    // Sets the pad configuration for the specified pin(s).

ROM_GPIOPinIntClear
    // Clears the interrupt for the specified pin(s).

ROM_GPIOPinIntDisable
    // Disables interrupts for the specified pin(s).

ROM_GPIOPinIntEnable
    // Enables interrupts for the specified pin(s).

ROM_GPIOPinIntStatus
    // Gets interrupt status for the specified GPIO port.

ROM_GPIOPinRead
    // Reads the values present of the specified pin(s).

ROM_GPIOPinTypeCAN
    // Configures pin(s) for use as a CAN device.

ROM_GPIOPinTypeComparator
    // Configures pin(s) for use as an analog comparator input.

ROM_GPIOPinTypeGPIOInput
    // Configures pin(s) for use as GPIO inputs.

ROM_GPIOPinTypeGPIOOutput
    // Configures pin(s) for use as GPIO outputs.

ROM_GPIOPinTypeGPIOOutputOD
    // Configures pin(s) for use as GPIO open drain outputs.

ROM_GPIOPinTypeI2C
    // Configures pin(s) for use by the I2C peripheral.

ROM_GPIOPinTypePWM
    // Configures pin(s) for use by the PWM peripheral.

ROM_GPIOPinTypeQEI
    // Configures pin(s) for use by the QEI peripheral.

ROM_GPIOPinTypeSSI
    // Configures pin(s) for use by the SSI peripheral.

ROM_GPIOPinTypeTimer
    // Configures pin(s) for use by the Timer peripheral.

ROM_GPIOPinTypeUART
    // Configures pin(s) for use by the UART peripheral.

ROM_GPIOPinWrite
    // Writes a value to the specified pin(s).

ROM_I2CMasterBusBusy
    // Indicates whether or not the I2C bus is busy.

ROM_I2CMasterBusy
    // Indicates whether or not the I2C Master is busy.

ROM_I2CMasterControl
    // Controls the state of the I2C Master module.

ROM_I2CMasterDataGet
    // Receives a byte that has been sent to the I2C Master.

ROM_I2CMasterDataPut
    // Transmits a byte from the I2C Master.

ROM_I2CMasterDisable
    // Disables the I2C master block.

ROM_I2CMasterEnable
    // Enables the I2C Master block.

ROM_I2CMasterErr
    // Gets the error status of the I2C Master module.

ROM_I2CMasterInitExpClk
    // Initializes the I2C Master block.

ROM_I2CMasterIntClear
    // Clears I2C Master interrupt sources.

ROM_I2CMasterIntDisable
    // Disables the I2C Master interrupt.

ROM_I2CMasterIntEnable
    // Enables the I2C Master interrupt.

ROM_I2CMasterIntStatus
    // Gets the current I2C Master interrupt status.

ROM_I2CMasterSlaveAddrSet
    // Sets the address that the I2C Master will place on the bus.

ROM_I2CSlaveDataGet
    // Receives a byte that has been sent to the I2C Slave.

ROM_I2CSlaveDataPut
    // Transmits a byte from the I2C Slave.

ROM_I2CSlaveDisable
    // Disables the I2C slave block.

ROM_I2CSlaveEnable
    // Enables the I2C Slave block.

ROM_I2CSlaveInit
    // Initializes the I2C Slave block.

ROM_I2CSlaveIntClear
    // Clears I2C Slave interrupt sources.

ROM_I2CSlaveIntDisable
    // Disables the I2C Slave interrupt.

ROM_I2CSlaveIntEnable
    // Enables the I2C Slave interrupt.

ROM_I2CSlaveIntStatus
    // Gets the current I2C Slave interrupt status.

ROM_I2CSlaveStatus
    // Gets the I2C Slave module status.

ROM_IntDisable
    // Disables an interrupt.

ROM_IntEnable
    // Enables an interrupt.

ROM_IntPriorityGet
    // Gets the priority of an interrupt.

ROM_IntPriorityGroupingGet
    // Gets the priority grouping of the interrupt controller.

ROM_IntPriorityGroupingSet
    // Sets the priority grouping of the interrupt controller.

ROM_IntPrioritySet
    // Sets the priority of an interrupt.

ROM_PWMDeadBandDisable
    // Disables the PWM dead band output.

ROM_PWMDeadBandEnable
    // Enables the PWM dead band output, and sets the dead band delays.

ROM_PWMFaultIntClear
    // Clears the fault interrupt for a PWM module.

ROM_PWMGenConfigure
    // Configures a PWM generator.

ROM_PWMGenDisable
    // Disables the timer/counter for a PWM generator block.

ROM_PWMGenEnable
    // Enables the timer/counter for a PWM generator block.

ROM_PWMGenIntClear
    // Clears the specified interrupt(s) for the specified PWM generator block.

ROM_PWMGenIntStatus
    // Gets interrupt status for the specified PWM generator block.

ROM_PWMGenIntTrigDisable
    // Disables interrupts for the specified PWM generator block.

ROM_PWMGenIntTrigEnable
    // Enables interrupts and triggers for the specified PWM generator block.

ROM_PWMGenPeriodGet
    // Gets the period of a PWM generator block.

ROM_PWMGenPeriodSet
    // Set the period of a PWM generator.

ROM_PWMIntDisable
    // Disables generator and fault interrupts for a PWM module.

ROM_PWMIntEnable
    // Enables generator and fault interrupts for a PWM module.

ROM_PWMIntStatus
    // Gets the interrupt status for a PWM module.

ROM_PWMOutputFault
    // Specifies the state of PWM outputs in response to a fault condition.

ROM_PWMOutputInvert
    // Selects the inversion mode for PWM outputs.

ROM_PWMOutputState
    // Enables or disables PWM outputs.

ROM_PWMPulseWidthGet
    // Gets the pulse width of a PWM output.

ROM_PWMPulseWidthSet
    // Sets the pulse width for the specified PWM output.

ROM_PWMSyncTimeBase
    // Synchronizes the counters in one or multiple PWM generator blocks.

ROM_PWMSyncUpdate
    // Synchronizes all pending updates.

ROM_QEIConfigure
    // Configures the quadrature encoder.

ROM_QEIDirectionGet
    // Gets the current direction of rotation.

ROM_QEIDisable
    // Disables the quadrature encoder.

ROM_QEIEnable
    // Enables the quadrature encoder.

ROM_QEIErrorGet
    // Gets the encoder error indicator.

ROM_QEIIntClear
    // Clears quadrature encoder interrupt sources.

ROM_QEIIntDisable
    // Disables individual quadrature encoder interrupt sources.

ROM_QEIIntEnable
    // Enables individual quadrature encoder interrupt sources.

ROM_QEIIntStatus
    // Gets the current interrupt status.

ROM_QEIPositionGet
    // Gets the current encoder position.

ROM_QEIPositionSet
    // Sets the current encoder position.

ROM_QEIVelocityConfigure
    // Configures the velocity capture.

ROM_QEIVelocityDisable
    // Disables the velocity capture.

ROM_QEIVelocityEnable
    // Enables the velocity capture.

ROM_QEIVelocityGet
    // Gets the current encoder speed.

ROM_SSIConfigSetExpClk
    // Configures the synchronous serial interface.

ROM_SSIDataGet
    // Gets a data element from the SSI receive FIFO.

ROM_SSIDataGetNonBlocking
    // Gets a data element from the SSI receive FIFO.

ROM_SSIDataPut
    // Puts a data element into the SSI transmit FIFO.

ROM_SSIDataPutNonBlocking
    // Puts a data element into the SSI transmit FIFO.

ROM_SSIDisable
    // Disables the synchronous serial interface.

ROM_SSIEnable
    // Enables the synchronous serial interface.

ROM_SSIIntClear
    // Clears SSI interrupt sources.

ROM_SSIIntDisable
    // Disables individual SSI interrupt sources.

ROM_SSIIntEnable
    // Enables individual SSI interrupt sources.

ROM_SSIIntStatus
    // Gets the current interrupt status.

ROM_SysCtlADCSpeedGet
    // Gets the sample rate of the ADC.

ROM_SysCtlADCSpeedSet
    // Sets the sample rate of the ADC.

ROM_SysCtlClockGet
    // Gets the processor clock rate.

ROM_SysCtlClockSet
    // Sets the clocking of the device.

ROM_SysCtlDeepSleep
    // Puts the processor into deep-sleep mode.

ROM_SysCtlFlashSizeGet
    // Gets the size of the flash.

ROM_SysCtlGPIOAHBDisable
    // Disables a GPIO peripheral for access from the high-speed bus.

ROM_SysCtlGPIOAHBEnable
    // Enables a GPIO peripheral for access from the high-speed bus.

ROM_SysCtlIntClear
    // Clears system control interrupt sources.

ROM_SysCtlIntDisable
    // Disables individual system control interrupt sources.

ROM_SysCtlIntEnable
    // Enables individual system control interrupt sources.

ROM_SysCtlIntStatus
    // Gets the current interrupt status.

ROM_SysCtlLDOGet
    // Gets the output voltage of the LDO.

ROM_SysCtlLDOSet
    // Sets the output voltage of the LDO.

ROM_SysCtlPeripheralClockGating
    // Controls peripheral clock gating in sleep and deep-sleep mode.

ROM_SysCtlPeripheralDeepSleepDisable
    // Disables a peripheral in deep-sleep mode.

ROM_SysCtlPeripheralDeepSleepEnable
    // Enables a peripheral in deep-sleep mode.

ROM_SysCtlPeripheralDisable
    // Disables a peripheral.

ROM_SysCtlPeripheralEnable
    // Enables a peripheral.

ROM_SysCtlPeripheralPresent
    // Determines if a peripheral is present.

ROM_SysCtlPeripheralReset
    // Performs a software reset of a peripheral.

ROM_SysCtlPeripheralSleepDisable
    // Disables a peripheral in sleep mode.

ROM_SysCtlPeripheralSleepEnable
// Enables a peripheral in sleep mode.

ROM_SysCtlPinPresent
// Determines if a pin is present.

ROM_SysCtlPWMClockGet
// Gets the current PWM clock configuration.

ROM_SysCtlPWMClockSet
// Sets the PWM clock configuration.

ROM_SysCtlReset
// Resets the device.

ROM_SysCtlResetCauseClear
// Clears reset reasons.

ROM_SysCtlResetCauseGet
// Gets the reason for a reset.

ROM_SysCtlSleep
// Puts the processor into sleep mode.

ROM_SysCtlSRAMSizeGet
// Gets the size of the SRAM.

ROM_SysTickDisable
// Disables the SysTick counter.

ROM_SysTickEnable
// Enables the SysTick counter.

ROM_SysTickIntDisable
// Disables the SysTick interrupt.

ROM_SysTickIntEnable
// Enables the SysTick interrupt.

ROM_SysTickPeriodGet
// Gets the period of the SysTick counter.

ROM_SysTickPeriodSet
// Sets the period of the SysTick counter.

ROM_SysTickValueGet
// Gets the current value of the SysTick counter.

ROM_TimerConfigure
// Configures the timer(s).

ROM_TimerControlEvent
// Controls the event type.

ROM_TimerControlLevel
    // Controls the output level.

ROM_TimerControlStall
    // Controls the stall handling.

ROM_TimerControlTrigger
    // Enables or disables the trigger output.

ROM_TimerDisable
    // Disables the timer(s).

ROM_TimerEnable
    // Enables the timer(s).

ROM_TimerIntClear
    // Clears timer interrupt sources.

ROM_TimerIntDisable
    // Disables individual timer interrupt sources.

ROM_TimerIntEnable
    // Enables individual timer interrupt sources.

ROM_TimerIntStatus
    // Gets the current interrupt status.

ROM_TimerLoadGet
    // Gets the timer load value.

ROM_TimerLoadSet
    // Sets the timer load value.

ROM_TimerMatchGet
    // Gets the timer match value.

ROM_TimerMatchSet
    // Sets the timer match value.

ROM_TimerPrescaleGet
    // Get the timer prescale value.

ROM_TimerPrescaleSet
    // Set the timer prescale value.

ROM_TimerRTCDisable
    // Disable RTC counting.

ROM_TimerRTCEnable
    // Enable RTC counting.

ROM_TimerValueGet
    // Gets the current timer value.

ROM_UARTBreakCtl
    // Causes a BREAK to be sent.

ROM_UARTCharGet
    // Waits for a character from the specified port.

ROM_UARTCharGetNonBlocking
    // Receives a character from the specified port.

ROM_UARTCharPut
    // Waits to send a character from the specified port.

ROM_UARTCharPutNonBlocking
    // Sends a character to the specified port.

ROM_UARTCharsAvail
    // Determines if there are any characters in the receive FIFO.

ROM_UARTConfigGetExpClk
    // Gets the current configuration of a UART.

ROM_UARTConfigSetExpClk
    // Sets the configuration of a UART.

ROM_UARTDisable
    // Disables transmitting and receiving.

ROM_UARTDisableSIR
    // Disables SIR (IrDA) mode on the specified UART.

ROM_UARTEnable
    // Enables transmitting and receiving.

ROM_UARTEnableSIR
    // Enables SIR (IrDA) mode on specified UART.

ROM_UARTFIFOLevelGet
    // Gets the FIFO level at which interrupts are generated.

ROM_UARTFIFOLevelSet
    // Sets the FIFO level at which interrupts are generated.

ROM_UARTIntClear
    // Clears UART interrupt sources.

ROM_UARTIntDisable
    // Disables individual UART interrupt sources.

ROM_UARTIntEnable
    // Enables individual UART interrupt sources.

ROM_UARTIntStatus
    // Gets the current interrupt status.

ROM_UARTParityModeGet
    // Gets the type of parity currently being used.

ROM_UARTParityModeSet
    // Sets the type of parity.

ROM_UARTSpaceAvail
    // Determines if there is any space in the transmit FIFO.

ROM_UpdateI2C
    // Starts an update over the I2C0 interface.

ROM_UpdateSSI
    // Starts an update over the SSI0 interface.

ROM_UpdateUART
    // Starts an update over the UART0 interface.

ROM_WatchdogEnable
    // Enables the watchdog timer.

ROM_WatchdogIntClear
    // Clears the watchdog timer interrupt.

ROM_WatchdogIntEnable
    // Enables the watchdog timer interrupt.

ROM_WatchdogIntStatus
    // Gets the current watchdog timer interrupt status.

ROM_WatchdogLock
    // Enables the watchdog timer lock mechanism.

ROM_WatchdogLockState
    // Gets the state of the watchdog timer lock mechanism.

ROM_WatchdogReloadGet
    // Gets the watchdog timer reload value.

ROM_WatchdogReloadSet
    // Sets the watchdog timer reload value.

ROM_WatchdogResetDisable
    // Disables the watchdog timer reset.

ROM_WatchdogResetEnable
    // Enables the watchdog timer reset.

ROM_WatchdogRunning
    // Determines if the watchdog timer is enabled.

ROM_WatchdogStallDisable
    // Disables stalling of the watchdog timer during debug events.

ROM_WatchdogStallEnable
    // Enables stalling of the watchdog timer during debug events.

ROM_WatchdogUnlock
    // Disables the watchdog timer lock mechanism.

ROM_WatchdogValueGet
    // Gets the current watchdog timer value.

C Register Quick Reference

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
System Control Base 0x400F.E000															
DID0, type RO, offset 0x000, reset -															
	VER							CLASS							
MAJOR								MINOR							
PBORCTL, type R/W, offset 0x030, reset 0x0000.7FFD															
														BORIOR	
LDOPCTL, type R/W, offset 0x034, reset 0x0000.0000															
										VADJ					
RIS, type RO, offset 0x050, reset 0x0000.0000															
							MOSCPUPRIS	USBPLLLRIS	PLLLRIS					BORRIS	
IMC, type R/W, offset 0x054, reset 0x0000.0000															
							MOSCPUPIM	USBPLLLIM	PLLLIM					BORIM	
MISC, type R/W1C, offset 0x058, reset 0x0000.0000															
							MOSCPUPMIS	USBPLLLMIS	PLLLMIS					BORMIS	
RESC, type R/W, offset 0x05C, reset -															
															MOSCFAIL
											SW	WDT	BOR	POR	EXT
RCC, type R/W, offset 0x060, reset 0x078E.3AD1															
				ACG	SYSDIV				USESYSDIV		USEPWMDIV	PWMDIV			
		PWRDN		BYPASS	XTAL					OSCSRC				IOSCDIS	MOSCDIS
PLLCFG, type RO, offset 0x064, reset -															
		F									R				
GPIOHSCTL, type R/W, offset 0x06C, reset 0x0000.0000															
								PORTHHS	PORTGHS	PORTFHS	PORTEHS	PORTDHS	PORTCHS	PORTBHS	PORTAHS
RCC2, type R/W, offset 0x070, reset 0x0780.6810															
USERCC2				SYSDIV2											
	USBPWRDN	PWRDN2		BYPASS2					OSCSRC2						
MOSCCTL, type R/W, offset 0x07C, reset 0x0000.0000															
															CVAL
DSLPCLKCFG, type R/W, offset 0x144, reset 0x0780.0000															
				DSDIVORIDE											
									DSOSCSRC						
DID1, type RO, offset 0x004, reset -															
VER				FAM				PARTNO							
PINCOUNT								TEMP			PKG	ROHS	QUAL		
DC0, type RO, offset 0x008, reset 0x00FF.003F															
SRAMSZ															
FLASHSZ															
DC1, type RO, offset 0x010, reset 0x0311.33FF															
						CAN1	CAN0				PWM				ADC
MINSYSDIV						MAXADCSPD		MPU	HIB	TEMPSNS	PLL	WDT	SWO	SWD	JTAG

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
DC2, type RO, offset 0x014, reset 0x030F.5133															
						COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
	I2C1		I2C0				QEI0			SSI1	SSI0			UART1	UART0
DC3, type RO, offset 0x018, reset 0x9FFF.8FFF															
32KHZ			CCP4	CCP3	CCP2	CCP1	CCP0	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
PWMFAULT				C1O	C1PLUS	C1MINUS	C0O	C0PLUS	C0MINUS	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0
DC4, type RO, offset 0x01C, reset 0x0000.30FF															
		UDMA	ROM					GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
DC5, type RO, offset 0x020, reset 0x0F30.00FF															
				PWMFAULT3	PWMFAULT2	PWMFAULT1	PWMFAULT0			PWMEFLT	PWMESYNC				
								PWM7	PWM6	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0
DC6, type RO, offset 0x024, reset 0x0000.0002															
														USB0	
DC7, type RO, offset 0x028, reset 0x03C0.0F3F															
						SSI1_TX	SSI1_RX	UART1_TX	UART1_RX						
				SSI0_TX	SSI0_RX	UART0_TX	UART0_RX			USB_EP3_TX	USB_EP3_RX	USB_EP2_TX	USB_EP2_RX	USB_EP1_TX	USB_EP1_RX
RCGC0, type R/W, offset 0x100, reset 0x00000040															
						CAN1	CAN0				PWM				ADC
						MAXADCSPD			HIB			WDT			
SCGC0, type R/W, offset 0x110, reset 0x00000040															
						CAN1	CAN0				PWM				ADC
						MAXADCSPD			HIB			WDT			
DCGC0, type R/W, offset 0x120, reset 0x00000040															
						CAN1	CAN0				PWM				ADC
						MAXADCSPD			HIB			WDT			
RCGC1, type R/W, offset 0x104, reset 0x00000000															
						COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
	I2C1		I2C0				QEI0			SSI1	SSI0			UART1	UART0
SCGC1, type R/W, offset 0x114, reset 0x00000000															
						COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
	I2C1		I2C0				QEI0			SSI1	SSI0			UART1	UART0
DCGC1, type R/W, offset 0x124, reset 0x00000000															
						COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
	I2C1		I2C0				QEI0			SSI1	SSI0			UART1	UART0
RCGC2, type R/W, offset 0x108, reset 0x00000000															
															USB0
		UDMA						GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
SCGC2, type R/W, offset 0x118, reset 0x00000000															
															USB0
		UDMA						GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
DCGC2, type R/W, offset 0x128, reset 0x00000000															
															USB0
		UDMA						GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
SRCR0, type R/W, offset 0x040, reset 0x00000000															
						CAN1	CAN0				PWM				ADC
									HIB			WDT			
SRCR1, type R/W, offset 0x044, reset 0x00000000															
						COMP1	COMP0					TIMER3	TIMER2	TIMER1	TIMER0
	I2C1		I2C0				QEI0			SSI1	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
SRCR2, type R/W, offset 0x048, reset 0x00000000															
															USB0
		UDMA						GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Hibernation Module Base 0x400F.C000															
HIBRTCC, type RO, offset 0x000, reset 0x0000.0000															
RTCC															
RTCC															
HIBRTCM0, type R/W, offset 0x004, reset 0xFFFF.FFFF															
RTCM0															
RTCM0															
HIBRTCM1, type R/W, offset 0x008, reset 0xFFFF.FFFF															
RTCM1															
RTCM1															
HIBRTCLD, type R/W, offset 0x00C, reset 0xFFFF.FFFF															
RTCLD															
RTCLD															
HIBCTL, type R/W, offset 0x010, reset 0x0000.0000															
WRC															
								VABORT	CLK32EN	LOWBATEN	PINWEN	RTCWEN	CLKSEL	HIBREQ	RTCEN
HIBIM, type R/W, offset 0x014, reset 0x0000.0000															
												EXTW	LOWBAT	RTCALT1	RTCALT0
HIBRIS, type RO, offset 0x018, reset 0x0000.0000															
												EXTW	LOWBAT	RTCALT1	RTCALT0
HIBMIS, type RO, offset 0x01C, reset 0x0000.0000															
												EXTW	LOWBAT	RTCALT1	RTCALT0
HIBIC, type R/W1C, offset 0x020, reset 0x0000.0000															
												EXTW	LOWBAT	RTCALT1	RTCALT0
HIBRTCT, type R/W, offset 0x024, reset 0x0000.7FFF															
TRIM															
HIBDATA, type R/W, offset 0x030-0x12C, reset 0x0000.0000															
RTD															
RTD															
Internal Memory **ROM Registers (System Control Offset)** Base 0x400F.E000															
RMCTL, type R/W1C, offset 0x0F0, reset -															
															BA
Internal Memory **Flash Registers (Flash Control Offset)** Base 0x400F.D000															
FMA, type R/W, offset 0x000, reset 0x0000.0000															
															OFFSET
OFFSET															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMD, type R/W, offset 0x004, reset 0x0000.0000															
DATA															
DATA															
FMC, type R/W, offset 0x008, reset 0x0000.0000															
WRKEY															
												COMT	MERASE	ERASE	WRITE
FCRIS, type RO, offset 0x00C, reset 0x0000.0000															
														PRIS	ARIS
FCIM, type R/W, offset 0x010, reset 0x0000.0000															
														PMASK	AMASK
FCMISC, type R/W1C, offset 0x014, reset 0x0000.0000															
														PMISC	AMISC

Internal Memory

Flash Registers (System Control Offset)

Base 0x400F.E000

USECRL, type R/W, offset 0x140, reset 0x31															
								USEC							
RMVER, type RO, offset 0x0F4, reset 0x0000.0000															
CONT								SIZE							
VER								REV							
FMPRE0, type R/W, offset 0x130 and 0x200, reset 0xFFFF.FFFF															
READ_ENABLE															
READ_ENABLE															
FMPPE0, type R/W, offset 0x134 and 0x400, reset 0xFFFF.FFFF															
PROG_ENABLE															
PROG_ENABLE															
USER_DBG, type R/W, offset 0x1D0, reset 0xFFFF.FFFE															
NW	DATA														
DATA														DBG1	DBG0
USER_REG0, type R/W, offset 0x1E0, reset 0xFFFF.FFFF															
NW	DATA														
DATA															
USER_REG1, type R/W, offset 0x1E4, reset 0xFFFF.FFFF															
NW	DATA														
DATA															
USER_REG2, type R/W, offset 0x1E8, reset 0xFFFF.FFFF															
NW	DATA														
DATA															
USER_REG3, type R/W, offset 0x1EC, reset 0xFFFF.FFFF															
NW	DATA														
DATA															
FMPRE1, type R/W, offset 0x204, reset 0xFFFF.FFFF															
READ_ENABLE															
READ_ENABLE															
FMPRE2, type R/W, offset 0x208, reset 0x0000.0000															
READ_ENABLE															
READ_ENABLE															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMPRE3, type R/W, offset 0x20C, reset 0x0000.0000															
READ_ENABLE															
READ_ENABLE															
FMPPE1, type R/W, offset 0x404, reset 0xFFFF.FFFF															
PROG_ENABLE															
PROG_ENABLE															
FMPPE2, type R/W, offset 0x408, reset 0x0000.0000															
PROG_ENABLE															
PROG_ENABLE															
FMPPE3, type R/W, offset 0x40C, reset 0x0000.0000															
PROG_ENABLE															
PROG_ENABLE															

Micro Direct Memory Access (µDMA)

µDMA Channel Control Structure

Base n/a

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
DMASRCENDP, type R/W, offset 0x000, reset -															
ADDR															
ADDR															
DMADSTENDP, type R/W, offset 0x004, reset -															
ADDR															
ADDR															
DMACHCTL, type R/W, offset 0x008, reset -															
DSTINC		DSTSIZE		SRCINC		SRCSIZE								ARBSIZE	
ARBSIZE				XFERSIZE								NXTUSEBURST		XFERMODE	

Micro Direct Memory Access (µDMA)

µDMA Registers

Base 0x400F.F000

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
DMASTAT, type RO, offset 0x000, reset 0x001F.0000															
											DMACHANS				
								STATE							MASTEN
DMACFG, type WO, offset 0x004, reset -															
															MASTEN
DMACTLBASE, type R/W, offset 0x008, reset 0x0000.0000															
ADDR															
ADDR															
DMAALTBASE, type RO, offset 0x00C, reset 0x0000.0200															
ADDR															
ADDR															
DMAWAITSTAT, type RO, offset 0x010, reset 0x0000.0000															
WAITREQ[n]															
WAITREQ[n]															
DMASWREQ, type WO, offset 0x014, reset -															
SWREQ[n]															
SWREQ[n]															
DMAUSEBURSTSET, type RO, offset 0x018, reset 0x0000.0000 (Reads)															
SET[n]															
SET[n]															
DMAUSEBURSTSET, type WO, offset 0x018, reset 0x0000.0000 (Writes)															
SET[n]															
SET[n]															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAUSEBURSTCLR, type WO, offset 0x01C, reset -															
CLR[n]															
CLR[n]															
DMAREQMASKSET, type RO, offset 0x020, reset 0x0000.0000 (Reads)															
SET[n]															
SET[n]															
DMAREQMASKSET, type WO, offset 0x020, reset 0x0000.0000 (Writes)															
SET[n]															
SET[n]															
DMAREQMASKCLR, type WO, offset 0x024, reset -															
CLR[n]															
CLR[n]															
DMAENASET, type RO, offset 0x028, reset 0x0000.0000 (Reads)															
SET[n]															
SET[n]															
DMAENASET, type WO, offset 0x028, reset 0x0000.0000 (Writes)															
SET[n]															
SET[n]															
DMAENACLR, type WO, offset 0x02C, reset -															
CLR[n]															
CLR[n]															
DMAALTSET, type RO, offset 0x030, reset 0x0000.0000 (Reads)															
SET[n]															
SET[n]															
DMAALTSET, type WO, offset 0x030, reset 0x0000.0000 (Writes)															
SET[n]															
SET[n]															
DMAALTCLR, type WO, offset 0x034, reset -															
CLR[n]															
CLR[n]															
DMAPRIOSET, type RO, offset 0x038, reset 0x0000.0000 (Reads)															
SET[n]															
SET[n]															
DMAPRIOSET, type WO, offset 0x038, reset 0x0000.0000 (Writes)															
SET[n]															
SET[n]															
DMAPRIOCLR, type WO, offset 0x03C, reset -															
CLR[n]															
CLR[n]															
DMAERRCLR, type RO, offset 0x04C, reset 0x0000.0000 (Reads)															
															ERRCLR
DMAERRCLR, type WO, offset 0x04C, reset 0x0000.0000 (Writes)															
															ERRCLR
DMAPeriphID0, type RO, offset 0xFE0, reset 0x0000.0030															
								PID0							
DMAPeriphID1, type RO, offset 0xFE4, reset 0x0000.00B2															
								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
DMAPeriphID2, type RO, offset 0xFE8, reset 0x0000.000B															
								PID2							
DMAPeriphID3, type RO, offset 0xFEC, reset 0x0000.0000															
								PID3							
DMAPeriphID4, type RO, offset 0xFD0, reset 0x0000.0004															
								PID4							
DMAPCellID0, type RO, offset 0xFF0, reset 0x0000.000D															
								CID0							
DMAPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0															
								CID1							
DMAPCellID2, type RO, offset 0xFF8, reset 0x0000.0005															
								CID2							
DMAPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1															
								CID3							
General-Purpose Input/Outputs (GPIOs) GPIO Port A (legacy) base: 0x4000.4000 GPIO Port A (high-speed) base: 0x4005.8000 GPIO Port B (legacy) base: 0x4000.5000 GPIO Port B (high-speed) base: 0x4005.9000 GPIO Port C (legacy) base: 0x4000.6000 GPIO Port C (high-speed) base: 0x4005.A000 GPIO Port D (legacy) base: 0x4000.7000 GPIO Port D (high-speed) base: 0x4005.B000 GPIO Port E (legacy) base: 0x4002.4000 GPIO Port E (high-speed) base: 0x4005.C000 GPIO Port F (legacy) base: 0x4002.5000 GPIO Port F (high-speed) base: 0x4005.D000 GPIO Port G (legacy) base: 0x4002.6000 GPIO Port G (high-speed) base: 0x4005.E000 GPIO Port H (legacy) base: 0x4002.7000 GPIO Port H (high-speed) base: 0x4005.F000															
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							

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
GPIORIS, type RO, offset 0x414, reset 0x0000.0000															
								RIS							
GPIOMIS, type RO, offset 0x418, reset 0x0000.0000															
								MIS							
GPIOICR, type W1C, offset 0x41C, reset 0x0000.0000															
								IC							
GPIOAFSEL, type R/W, offset 0x420, reset -															
								AFSEL							
GPIODR2R, type R/W, offset 0x500, reset 0x0000.00FF															
								DRV2							
GPIODR4R, type R/W, offset 0x504, reset 0x0000.0000															
								DRV4							
GPIODR8R, type R/W, offset 0x508, reset 0x0000.0000															
								DRV8							
GPIOODR, type R/W, offset 0x50C, reset 0x0000.0000															
								ODE							
GPIOPUR, type R/W, offset 0x510, reset -															
								PUE							
GPIOPDR, type R/W, offset 0x514, reset 0x0000.0000															
								PDE							
GPIOSLR, type R/W, offset 0x518, reset 0x0000.0000															
								SRL							
GPIODEN, type R/W, offset 0x51C, reset -															
								DEN							
GPIOLOCK, type R/W, offset 0x520, reset 0x0000.0001															
								LOCK							
								LOCK							
GPIOCR, type -, offset 0x524, reset -															
								CR							
GPIOAMSEL, type R/W, offset 0x528, reset 0x0000.0000															
								GPIOAMSEL							
GPIOPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000															
								PID4							
GPIOPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000															
								PID5							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000															
								PID6							
GPIOPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000															
								PID7							
GPIOPeriphID0, type RO, offset 0xFE0, reset 0x0000.0061															
								PID0							
GPIOPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000															
								PID1							
GPIOPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018															
								PID2							
GPIOPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001															
								PID3							
GPIOPCellID0, type RO, offset 0xFF0, reset 0x0000.000D															
								CID0							
GPIOPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0															
								CID1							
GPIOPCellID2, type RO, offset 0xFF8, reset 0x0000.0005															
								CID2							
GPIOPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1															
								CID3							
General-Purpose Timers Timer0 base: 0x4003.0000 Timer1 base: 0x4003.1000 Timer2 base: 0x4003.2000 Timer3 base: 0x4003.3000															
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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPTMMIS, type RO, offset 0x020, reset 0x0000.0000															
					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															
GPTMTBMATCHR, type R/W, offset 0x034, reset 0x0000.FFFF															
TBMRL															
GPTMTAPR, type R/W, offset 0x038, reset 0x0000.0000															
								TAPSR							
GPTMTBPR, type R/W, offset 0x03C, reset 0x0000.0000															
								TBPSR							
GPTMTAR, type RO, offset 0x048, reset 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								

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
WDTLOCK, type R/W, offset 0xC00, reset 0x0000.0000															
WDTLock															
WDTLock															
WDTPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000															
PID4															
WDTPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000															
PID5															
WDTPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000															
PID6															
WDTPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000															
PID7															
WDTPeriphID0, type RO, offset 0xFE0, reset 0x0000.0005															
PID0															
WDTPeriphID1, type RO, offset 0xFE4, reset 0x0000.0018															
PID1															
WDTPeriphID2, type RO, offset 0xFE8, reset 0x0000.0018															
PID2															
WDTPeriphID3, type RO, offset 0xFEC, reset 0x0000.0001															
PID3															
WDTPCellID0, type RO, offset 0xFF0, reset 0x0000.000D															
CID0															
WDTPCellID1, type RO, offset 0xFF4, reset 0x0000.00F0															
CID1															
WDTPCellID2, type RO, offset 0xFF8, reset 0x0000.0005															
CID2															
WDTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1															
CID3															
Analog-to-Digital Converter (ADC) Base 0x4003.8000															
ADCACTSS, type R/W, offset 0x000, reset 0x0000.0000															
												ASEN3	ASEN2	ASEN1	ASEN0
ADCRIS, type RO, offset 0x004, reset 0x0000.0000															
												INR3	INR2	INR1	INR0
ADCIM, type R/W, offset 0x008, reset 0x0000.0000															
												MASK3	MASK2	MASK1	MASK0
ADCISC, type R/W1C, offset 0x00C, reset 0x0000.0000															
												IN3	IN2	IN1	IN0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCOSTAT, type R/W1C, offset 0x010, reset 0x0000.0000															
												OV3	OV2	OV1	OV0
ADCEMUX, type R/W, offset 0x014, reset 0x0000.0000															
EM3				EM2				EM1				EM0			
ADCUSTAT, type R/W1C, offset 0x018, reset 0x0000.0000															
												UV3	UV2	UV1	UV0
ADCSSPRI, type R/W, offset 0x020, reset 0x0000.3210															
		SS3				SS2				SS1				SS0	
ADCPSSI, type WO, offset 0x028, reset -															
												SS3	SS2	SS1	SS0
ADCSAC, type R/W, offset 0x030, reset 0x0000.0000															
													AVG		
ADCSSMUX0, type R/W, offset 0x040, reset 0x0000.0000															
	MUX7				MUX6				MUX5				MUX4		
	MUX3				MUX2				MUX1				MUX0		
ADCSSCTL0, type R/W, offset 0x044, reset 0x0000.0000															
TS7	IE7	END7	D7	TS6	IE6	END6	D6	TS5	IE5	END5	D5	TS4	IE4	END4	D4
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
ADCSSFIFO0, type RO, offset 0x048, reset 0x0000.0000															
						DATA									
ADCSSFIFO1, type RO, offset 0x068, reset 0x0000.0000															
						DATA									
ADCSSFIFO2, type RO, offset 0x088, reset 0x0000.0000															
						DATA									
ADCSSFIFO3, type RO, offset 0x0A8, reset 0x0000.0000															
						DATA									
ADCSSFSTAT0, type RO, offset 0x04C, reset 0x0000.0100															
			FULL				EMPTY			HPTR				TPTR	
ADCSSFSTAT1, type RO, offset 0x06C, reset 0x0000.0100															
			FULL				EMPTY			HPTR				TPTR	
ADCSSFSTAT2, type RO, offset 0x08C, reset 0x0000.0100															
			FULL				EMPTY			HPTR				TPTR	
ADCSSFSTAT3, type RO, offset 0x0AC, reset 0x0000.0100															
			FULL				EMPTY			HPTR				TPTR	
ADCSSMUX1, type R/W, offset 0x060, reset 0x0000.0000															
	MUX3				MUX2				MUX1				MUX0		

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
ADCSSMUX2, type R/W, offset 0x080, reset 0x0000.0000															
	MUX3				MUX2				MUX1				MUX0		
ADCSSCTL1, type R/W, offset 0x064, reset 0x0000.0000															
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
ADCSSCTL2, type R/W, offset 0x084, reset 0x0000.0000															
TS3	IE3	END3	D3	TS2	IE2	END2	D2	TS1	IE1	END1	D1	TS0	IE0	END0	D0
ADCSSMUX3, type R/W, offset 0x0A0, reset 0x0000.0000															
														MUX0	
ADCSSCTL3, type R/W, offset 0x0A4, reset 0x0000.0002															
												TS0	IE0	END0	D0
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 (Reads)															
												OE	BE	PE	FE
UARTRSR/UARTECR, type WO, offset 0x004, reset 0x0000.0000 (Writes)															
								DATA							
UARTFR, type RO, offset 0x018, reset 0x0000.0090															
								TXFE	RXFF	TXFF	RXFE	BUSY			
UARTILPR, type R/W, offset 0x020, reset 0x0000.0000															
								ILPDVSR							
UARTIBRD, type R/W, offset 0x024, reset 0x0000.0000															
DIVINT															
UARTFBRD, type R/W, offset 0x028, reset 0x0000.0000															
										DIVFRAC					
UARTLCRH, type R/W, offset 0x02C, reset 0x0000.0000															
								SPS	WLEN		FEN	STP2	EPS	PEN	BRK
UARTCTL, type R/W, offset 0x030, reset 0x0000.0300															
						RXE	TXE	LBE					SIRLP	SIREN	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				

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
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				
UARTDMACTL, type R/W, offset 0x048, reset 0x0000.0000															
													DMAERR	TXDMAE	RXDMAE
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							
UARTPCellID2, type RO, offset 0xFF8, reset 0x0000.0005															
								CID2							
UARTPCellID3, type RO, offset 0xFFC, reset 0x0000.00B1															
								CID3							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Synchronous Serial Interface (SSI) SSI0 base: 0x4000.8000 SSI1 base: 0x4000.9000															
SSICR0, type R/W, offset 0x000, reset 0x0000.0000															
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
SSIDMACTL, type R/W, offset 0x024, reset 0x0000.0000															
														TXDMAE	RXDMAE
SSIPeriphID4, type RO, offset 0xFD0, reset 0x0000.0000															
								PID4							
SSIPeriphID5, type RO, offset 0xFD4, reset 0x0000.0000															
								PID5							
SSIPeriphID6, type RO, offset 0xFD8, reset 0x0000.0000															
								PID6							
SSIPeriphID7, type RO, offset 0xFDC, reset 0x0000.0000															
								PID7							
SSIPeriphID0, type RO, offset 0xFE0, reset 0x0000.0022															
								PID0							
SSIPeriphID1, type RO, offset 0xFE4, reset 0x0000.0000															
								PID1							

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 I2C Master 1 base: 0x4002.1000															
I2CMSA, type R/W, offset 0x000, reset 0x0000.0000															
								SA							R/S
I2CMCS, type RO, offset 0x004, reset 0x0000.0000 (Reads)															
									BUSBSY	IDLE	ARBLST	DATACK	ADRACK	ERROR	BUSY
I2CMCS, type WO, offset 0x004, reset 0x0000.0000 (Writes)															
												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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Inter-Integrated Circuit (I^2C) Interface															
I^2C Slave															
I2C Slave 0 base: 0x4002.0800 I2C Slave 1 base: 0x4002.1800															
I2CSOAR, type R/W, offset 0x000, reset 0x0000.0000															
									OAR						
I2CSCSR, type RO, offset 0x004, reset 0x0000.0000 (Reads)															
													FBR	TREQ	RREQ
I2CSCSR, type WO, offset 0x004, reset 0x0000.0000 (Writes)															
															DA
I2CSDR, type R/W, offset 0x008, reset 0x0000.0000															
								DATA							
I2CSIMR, type R/W, offset 0x00C, reset 0x0000.0000															
													STOPIM	STARTIM	DATAIM
I2CSRIS, type RO, offset 0x010, reset 0x0000.0000															
													STOPRIS	STARTRIS	DATARIS
I2CSMIS, type RO, offset 0x014, reset 0x0000.0000															
													STOPMIS	STARTMIS	DATAMIS
I2CSICR, type WO, offset 0x018, reset 0x0000.0000															
													STOPIC	STARTIC	DATAIC
Controller Area Network (CAN) Module															
CAN0 base: 0x4004.0000 CAN1 base: 0x4004.1000															
CANCTL, type R/W, offset 0x000, reset 0x0000.0001															
								TEST	CCE	DAR		EIE	SIE	IE	INIT
CANSTS, type R/W, offset 0x004, reset 0x0000.0000															
								BOFF	EWARN	EPASS	RXOK	TXOK	LEC		
CANERR, type RO, offset 0x008, reset 0x0000.0000															
RP	REC							TEC							
CANBIT, type R/W, offset 0x00C, reset 0x0000.2301															
	TSEG2			TSEG1				SJW		BRP					
CANINT, type RO, offset 0x010, reset 0x0000.0000															
INTID															
CANTST, type R/W, offset 0x014, reset 0x0000.0000															
								RX	TX		LBACK	SILENT	BASIC		
CANBRPE, type R/W, offset 0x018, reset 0x0000.0000															
												BRPE			

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
CANIF1CRQ, type R/W, offset 0x020, reset 0x0000.0001															
BUSY										MNUM					
CANIF2CRQ, type R/W, offset 0x080, reset 0x0000.0001															
BUSY										MNUM					
CANIF1CMSK, type R/W, offset 0x024, reset 0x0000.0000															
								WRNRD	MASK	ARB	CONTROL	CLRINTPND	NEWDAT / TXRQST	DATAA	DATAB
CANIF2CMSK, type R/W, offset 0x084, reset 0x0000.0000															
								WRNRD	MASK	ARB	CONTROL	CLRINTPND	NEWDAT / TXRQST	DATAA	DATAB
CANIF1MSK1, type R/W, offset 0x028, reset 0x0000.FFFF															
MSK															
CANIF2MSK1, type R/W, offset 0x088, reset 0x0000.FFFF															
MSK															
CANIF1MSK2, type R/W, offset 0x02C, reset 0x0000.FFFF															
MXTD	MDIR		MSK												
CANIF2MSK2, type R/W, offset 0x08C, reset 0x0000.FFFF															
MXTD	MDIR		MSK												
CANIF1ARB1, type R/W, offset 0x030, reset 0x0000.0000															
ID															
CANIF2ARB1, type R/W, offset 0x090, reset 0x0000.0000															
ID															
CANIF1ARB2, type R/W, offset 0x034, reset 0x0000.0000															
MSGVAL	XTD	DIR	ID												
CANIF2ARB2, type R/W, offset 0x094, reset 0x0000.0000															
MSGVAL	XTD	DIR	ID												
CANIF1MCTL, type R/W, offset 0x038, reset 0x0000.0000															
NEWDAT	MSGLST	INTPND	UMASK	TXIE	RXIE	RMTEN	TXRQST	EOB				DLC			
CANIF2MCTL, type R/W, offset 0x098, reset 0x0000.0000															
NEWDAT	MSGLST	INTPND	UMASK	TXIE	RXIE	RMTEN	TXRQST	EOB				DLC			
CANIF1DA1, type R/W, offset 0x03C, reset 0x0000.0000															
DATA															
CANIF1DA2, type R/W, offset 0x040, reset 0x0000.0000															
DATA															
CANIF1DB1, type R/W, offset 0x044, reset 0x0000.0000															
DATA															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CANIF1DB2, type R/W, offset 0x048, reset 0x0000.0000															
DATA															
CANIF2DA1, type R/W, offset 0x09C, reset 0x0000.0000															
DATA															
CANIF2DA2, type R/W, offset 0x0A0, reset 0x0000.0000															
DATA															
CANIF2DB1, type R/W, offset 0x0A4, reset 0x0000.0000															
DATA															
CANIF2DB2, type R/W, offset 0x0A8, reset 0x0000.0000															
DATA															
CANTXRQ1, type RO, offset 0x100, reset 0x0000.0000															
TXRQST															
CANTXRQ2, type RO, offset 0x104, reset 0x0000.0000															
TXRQST															
CANNWDA1, type RO, offset 0x120, reset 0x0000.0000															
NEWDAT															
CANNWDA2, type RO, offset 0x124, reset 0x0000.0000															
NEWDAT															
CANMSG1INT, type RO, offset 0x140, reset 0x0000.0000															
INTPND															
CANMSG2INT, type RO, offset 0x144, reset 0x0000.0000															
INTPND															
CANMSG1VAL, type RO, offset 0x160, reset 0x0000.0000															
MSGVAL															
CANMSG2VAL, type RO, offset 0x164, reset 0x0000.0000															
MSGVAL															
Universal Serial Bus (USB) Controller Base 0x4005.0000															
USBFADDR, type R/W, offset 0x000, reset 0x00															
									FUNCADDR						
USBPOWER, type R/W, offset 0x001, reset 0x20 (Host Mode)															
												RESET	RESUME	SUSPEND	PWRDNPHY
USBPOWER, type R/W, offset 0x001, reset 0x20 (Device Mode)															
								ISOUP	SOFTCONN			RESET	RESUME	SUSPEND	PWRDNPHY
USBTXIS, type RO, offset 0x002, reset 0x0000															
												EP3	EP2	EP1	EP0
USBRXIS, type RO, offset 0x004, reset 0x0000															
												EP3	EP2	EP1	

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBTXIE, type R/W, offset 0x006, reset 0x000F															
												EP3	EP2	EP1	EP0
USBRXIE, type R/W, offset 0x008, reset 0x000E															
												EP3	EP2	EP1	
USBIS, type RO, offset 0x00A, reset 0x00 (Host Mode)															
										DISCON	CONN	SOF	BABBLE	RESUME	
USBIS, type RO, offset 0x00A, reset 0x00 (Device Mode)															
										DISCON		SOF	RESET	RESUME	SUSPEND
USBIE, type R/W, offset 0x00B, reset 0x06 (Host Mode)															
										DISCON	CONN	SOF	RESET	RESUME	SUSPND
USBIE, type R/W, offset 0x00B, reset 0x06 (Device Mode)															
										DISCON	CONN	SOF	BABBLE	RESUME	SUSPND
USBFRAME, type RO, offset 0x00C, reset 0x0000															
					Frame										
USBEPIDX, type R/W, offset 0x0E, reset 0x0000															
												EPIDX			
USBTEST, type R/W, offset 0x00F, reset 0x00 (Host Mode)															
								FORCEH	FIFOACC	FORCEFS					
USBTEST, type R/W, offset 0x00F, reset 0x00 (Device Mode)															
									FIFOACC	FORCEFS					
USBFIFO0, type R/W, offset 0x020, reset 0x0000.0000															
							EPDATA								
							EPDATA								
USBFIFO1, type R/W, offset 0x024, reset 0x0000.0000															
							EPDATA								
							EPDATA								
USBFIFO2, type R/W, offset 0x028, reset 0x0000.0000															
							EPDATA								
							EPDATA								
USBFIFO3, type R/W, offset 0x02C, reset 0x0000.0000															
							EPDATA								
							EPDATA								
USBDEVCTL, type R/W, offset 0x060, reset 0x80 (Host Mode)															
								DEV	FSDEV	LSDEV			HOST		
USBDEVCTL, type R/W, offset 0x060, reset 0x80 (Device Mode)															
								DEV							
USBTXFIFOSZ, type R/W, offset 0x062, reset 0x00															
											DPB	SIZE			
USBRXFIFOSZ, type R/W, offset 0x063, reset 0x00															
											DPB	SIZE			
USBTXFIFOADD, type R/W, offset 0x064, reset 0x0000															
				ADDR											
USBRXFIFOADD, type R/W, offset 0x066, reset 0x0000															
				ADDR											
USBCONTIM, type R/W, offset 0x07A, reset 0x5C															
								WTCON							
USBFSEOF, type R/W, offset 0x07D, reset 0x77															
								FSEOFG							
USBLSEOF, type R/W, offset 0x07E, reset 0x72															
								LSEOFG							
USBTXFUNCADDR0, type R/W, offset 0x080, reset 0x00															
									ADDR						

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBTXFUNCADDR1, type R/W, offset 0x088, reset 0x00															
									ADDR						
USBTXFUNCADDR2, type R/W, offset 0x090, reset 0x00															
									ADDR						
USBTXFUNCADDR3, type R/W, offset 0x098, reset 0x00															
									ADDR						
USBTXHUBADDR0, type R/W, offset 0x082, reset 0x00															
								MULTTRAN	ADDR						
USBTXHUBADDR1, type R/W, offset 0x08A, reset 0x00															
								MULTTRAN	ADDR						
USBTXHUBADDR2, type R/W, offset 0x092, reset 0x00															
								MULTTRAN	ADDR						
USBTXHUBADDR3, type R/W, offset 0x09A, reset 0x00															
								MULTTRAN	ADDR						
USBTXHUBPORT0, type R/W, offset 0x083, reset 0x00															
									PORT						
USBTXHUBPORT1, type R/W, offset 0x08B, reset 0x00															
									PORT						
USBTXHUBPORT2, type R/W, offset 0x093, reset 0x00															
									PORT						
USBTXHUBPORT3, type R/W, offset 0x09B, reset 0x00															
									PORT						
USBRXFUNCADDR1, type R/W, offset 0x08C, reset 0x00															
									ADDR						
USBRXFUNCADDR2, type R/W, offset 0x094, reset 0x00															
									ADDR						
USBRXFUNCADDR3, type R/W, offset 0x09C, reset 0x00															
									ADDR						
USBRXHUBADDR1, type R/W, offset 0x08E, reset 0x00															
								MULTTRAN	ADDR						
USBRXHUBADDR2, type R/W, offset 0x096, reset 0x00															
								MULTTRAN	ADDR						
USBRXHUBADDR3, type R/W, offset 0x09E, reset 0x00															
								MULTTRAN	ADDR						
USBRXHUBPORT1, type R/W, offset 0x08F, reset 0x00															
									PORT						
USBRXHUBPORT2, type R/W, offset 0x097, reset 0x00															
									PORT						
USBRXHUBPORT3, type R/W, offset 0x09F, reset 0x00															
									PORT						
USBTXMAXP1, type R/W, offset 0x110, reset 0x0000															
MULT								MAXLOAD							
USBTXMAXP2, type R/W, offset 0x120, reset 0x0000															
MULT								MAXLOAD							
USBTXMAXP3, type R/W, offset 0x130, reset 0x0000															
MULT								MAXLOAD							
USBCSRL0, type W1C, offset 0x102, reset 0x00 (Host Mode)															
								NAKTO	STATUS	REQPKT	ERROR	SETUP	STALLED	TXRDY	RXRDY
USBCSRL0, type W1C, offset 0x102, reset 0x00 (Device Mode)															
								SETENDC	RXRDYC	STALL	SETEND	DATAEND	STALLED	TXRDY	RXRDY
USBCSRH0, type W1C, offset 0x103, reset 0x00 (Host Mode)															
													DTWE	DT	FLUSH

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
USBCSRH0, type W1C, offset 0x103, reset 0x00 (Device Mode)															
															FLUSH
USBCOUNT0, type RO, offset 0x108, reset 0x00															
									COUNT						
USBTYPE0, type R/W, offset 0x10A, reset 0x00															
								SPEED							
USBNAKLMT, type R/W, offset 0x10B, reset 0x00															
											NAKLMT				
USBTXCSRL1, type R/W, offset 0x112, reset 0x00 (Host Mode)															
								NAKTO / INCTX	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL2, type R/W, offset 0x122, reset 0x00 (Host Mode)															
								NAKTO / INCTX	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL3, type R/W, offset 0x132, reset 0x00 (Host Mode)															
								NAKTO / INCTX	CLRDT	STALLED	SETUP	FLUSH	ERROR	FIFONE	TXRDY
USBTXCSRL1, type R/W, offset 0x112, reset 0x00 (Device Mode)															
								INCTX	CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRL2, type R/W, offset 0x122, reset 0x00 (Device Mode)															
								INCTX	CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRL3, type R/W, offset 0x132, reset 0x00 (Device Mode)															
								INCTX	CLRDT	STALLED	STALL	FLUSH	UNDRN	FIFONE	TXRDY
USBTXCSRH1, type R/W, offset 0x113, reset 0x00 (Host Mode)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH2, type R/W, offset 0x123, reset 0x00 (Host Mode)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH3, type R/W, offset 0x133, reset 0x00 (Host Mode)															
								AUTOSET		MODE	DMAEN	FDT	DMAMOD	DTWE	DT
USBTXCSRH1, type R/W, offset 0x113, reset 0x00 (Device Mode)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH2, type R/W, offset 0x123, reset 0x00 (Device Mode)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBTXCSRH3, type R/W, offset 0x133, reset 0x00 (Device Mode)															
								AUTOSET	ISO	MODE	DMAEN	FDT	DMAMOD		
USBRXMAXP1, type R/W, offset 0x114, reset 0x0000															
MULT								MAXLOAD							
USBRXMAXP2, type R/W, offset 0x124, reset 0x0000															
MULT								MAXLOAD							
USBRXMAXP3, type R/W, offset 0x134, reset 0x0000															
MULT								MAXLOAD							
USBRXCSRL1, type R/W, offset 0x116, reset 0x00 (Host Mode)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL2, type R/W, offset 0x126, reset 0x00 (Host Mode)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL3, type R/W, offset 0x136, reset 0x00 (Host Mode)															
								CLRDT	STALLED	REQPKT	FLUSH	DATAERR / NAKTO	ERROR	FULL	RXRDY
USBRXCSRL1, type R/W, offset 0x116, reset 0x00 (Device Mode)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRL2, type R/W, offset 0x126, reset 0x00 (Device Mode)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBRXCSRL3, type R/W, offset 0x136, reset 0x00 (Device Mode)															
								CLRDT	STALLED	STALL	FLUSH	DATAERR	OVER	FULL	RXRDY
USBRXCSRH1, type R/W, offset 0x117, reset 0x00 (Host Mode)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	INCRX
USBRXCSRH2, type R/W, offset 0x127, reset 0x00 (Host Mode)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	INCRX
USBRXCSRH3, type R/W, offset 0x137, reset 0x00 (Host Mode)															
								AUTOCL	AUTORQ	DMAEN	PIDERR	DMAMOD	DTWE	DT	INCRX
USBRXCSRH1, type R/W, offset 0x117, reset 0x00 (Device Mode)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			INCRX
USBRXCSRH2, type R/W, offset 0x127, reset 0x00 (Device Mode)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			INCRX
USBRXCSRH3, type R/W, offset 0x137, reset 0x00 (Device Mode)															
								AUTOCL	ISO	DMAEN	DISNYET / PIDERR	DMAMOD			INCRX
USBRXCOUNT1, type RO, offset 0x118, reset 0x0000															
				COUNT											
USBRXCOUNT2, type RO, offset 0x128, reset 0x0000															
				COUNT											
USBRXCOUNT3, type RO, offset 0x138, reset 0x0000															
				COUNT											
USBTXTYPE1, type R/W, offset 0x11A, reset 0x00															
								SPEED		PROTO		TEP			
USBTXTYPE2, type R/W, offset 0x12A, reset 0x00															
								SPEED		PROTO		TEP			
USBTXTYPE3, type R/W, offset 0x13A, reset 0x00															
								SPEED		PROTO		TEP			
USBTXINTERVAL1, type R/W, offset 0x11B, reset 0x00															
								TXPOLL / NAKLMT							
USBTXINTERVAL2, type R/W, offset 0x12B, reset 0x00															
								TXPOLL / NAKLMT							
USBTXINTERVAL3, type R/W, offset 0x13B, reset 0x00															
								TXPOLL / NAKLMT							
USBRXTYPE1, type R/W, offset 0x11C, reset 0x00															
								SPEED		PROTO		TEP			
USBRXTYPE2, type R/W, offset 0x12C, reset 0x00															
								SPEED		PROTO		TEP			
USBRXTYPE3, type R/W, offset 0x13C, reset 0x00															
								SPEED		PROTO		TEP			
USBRXINTERVAL1, type R/W, offset 0x11D, reset 0x00															
								TXPOLL / NAKLMT							
USBRXINTERVAL2, type R/W, offset 0x12D, reset 0x00															
								TXPOLL / NAKLMT							
USBRXINTERVAL3, type R/W, offset 0x13D, reset 0x00															
								TXPOLL / NAKLMT							
USBRQPKTCOUNT1, type R/W, offset 0x304, reset 0x0000															
								COUNT							
USBRQPKTCOUNT2, type R/W, offset 0x308, reset 0x0000															
								COUNT							
USBRQPKTCOUNT3, type R/W, offset 0x30C, reset 0x0000															
								COUNT							

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
USBRXDPKTBUFDIS, type R/W, offset 0x340, reset 0x0000															
												EP3	EP2	EP1	
USBTXDPKTBUFDIS, type R/W, offset 0x342, reset 0x0000															
												EP3	EP2	EP1	
USBEPC, type R/W, offset 0x400, reset 0x0000.0000															
						PFLTACT			PFLTAEN	PFLTSEN	PFLTEN		EPENDE	EPEN	
USBEPCRIS, type RO, offset 0x404, reset 0x0000.0000															
															PF
USBEPCIM, type R/W, offset 0x408, reset 0x0000.0000															
															PF
USBEPCISC, type R/W, offset 0x40C, reset 0x0000.0000															
															PF
USBDRRIS, type RO, offset 0x410, reset 0x0000.0000															
															RESUME
USBDRIM, type R/W, offset 0x414, reset 0x0000.0000															
															RESUME
USBDRISC, type W1C, offset 0x418, reset 0x0000.0000															
															RESUME
USBGPCS, type R/W, offset 0x41C, reset 0x0000.0000															
															DEVMOD
Analog Comparators Base 0x4003.C000															
ACMIS, type R/W1C, offset 0x000, reset 0x0000.0000															
														IN1	IN0
ACRIS, type RO, offset 0x004, reset 0x0000.0000															
														IN1	IN0
ACINTEN, type R/W, offset 0x008, reset 0x0000.0000															
														IN1	IN0
ACREFCTL, type R/W, offset 0x010, reset 0x0000.0000															
						EN	RNG					VREF			
ACSTAT0, type RO, offset 0x020, reset 0x0000.0000															
														OVAL	
ACSTAT1, type RO, offset 0x040, reset 0x0000.0000															
														OVAL	
ACCTL0, type R/W, offset 0x024, reset 0x0000.0000															
				TOEN	ASRCP			TSLVAL	TSEN		ISLVAL	ISEN		CINV	

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 0x044, reset 0x0000.0000															
				TOEN	ASRCP			TSLVAL	TSEN		ISLVAL	ISEN		CINV	
Pulse Width Modulator (PWM) Base 0x4002.8000															
PWMCTL, type R/W, offset 0x000, reset 0x0000.0000															
												GlobalSync3	GlobalSync2	GlobalSync1	GlobalSync0
PWMSYNC, type R/W, offset 0x004, reset 0x0000.0000															
												Sync3	Sync2	Sync1	Sync0
PWMENABLE, type R/W, offset 0x008, reset 0x0000.0000															
								PWM7En	PWM6En	PWM5En	PWM4En	PWM3En	PWM2En	PWM1En	PWM0En
PWMINVERT, type R/W, offset 0x00C, reset 0x0000.0000															
								PWM7Inv	PWM6Inv	PWM5Inv	PWM4Inv	PWM3Inv	PWM2Inv	PWM1Inv	PWM0Inv
PWMFAULT, type R/W, offset 0x010, reset 0x0000.0000															
								Fault7	Fault6	Fault5	Fault4	Fault3	Fault2	Fault1	Fault0
PWMINTEN, type R/W, offset 0x014, reset 0x0000.0000															
												IntFault3	IntFault2	IntFault1	IntFault0
												IntPWM3	IntPWM2	IntPWM1	IntPWM0
PWMRIS, type RO, offset 0x018, reset 0x0000.0000															
												IntFault3	IntFault2	IntFault1	IntFault0
												IntPWM3	IntPWM2	IntPWM1	IntPWM0
PWMISC, type R/W1C, offset 0x01C, reset 0x0000.0000															
												IntFault3	IntFault2	IntFault1	IntFault0
												IntPWM3	IntPWM2	IntPWM1	IntPWM0
PWMSTATUS, type RO, offset 0x020, reset 0x0000.0000															
												Fault3	Fault2	Fault1	Fault0
PWMFAULTVAL, type R/W, offset 0x024, reset 0x0000.0000															
								PWM7	PWM6	PWM5	PWM4	PWM3	PWM2	PWM1	PWM0
PWM0CTL, type R/W, offset 0x040, reset 0x0000.0000															
													LATCH	MINFLTPER	FLTSRC
DBFallUpd		DBRiseUpd		DBCtlUpd		GenBUpd		GenAUpd		CmpBUpd	CmpAUpd	LoadUpd	Debug	Mode	Enable
PWM1CTL, type R/W, offset 0x080, reset 0x0000.0000															
													LATCH	MINFLTPER	FLTSRC
DBFallUpd		DBRiseUpd		DBCtlUpd		GenBUpd		GenAUpd		CmpBUpd	CmpAUpd	LoadUpd	Debug	Mode	Enable
PWM2CTL, type R/W, offset 0x0C0, reset 0x0000.0000															
													LATCH	MINFLTPER	FLTSRC
DBFallUpd		DBRiseUpd		DBCtlUpd		GenBUpd		GenAUpd		CmpBUpd	CmpAUpd	LoadUpd	Debug	Mode	Enable
PWM3CTL, type R/W, offset 0x100, reset 0x0000.0000															
													LATCH	MINFLTPER	FLTSRC
DBFallUpd		DBRiseUpd		DBCtlUpd		GenBUpd		GenAUpd		CmpBUpd	CmpAUpd	LoadUpd	Debug	Mode	Enable
PWM0INTEN, type R/W, offset 0x044, reset 0x0000.0000															
		TrCmpBD	TrCmpBU	TrCmpAD	TrCmpAU	TrCntLoad	TrCntZero			IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM1INTEN, type R/W, offset 0x084, reset 0x0000.0000															
		TrCmpBD	TrCmpBU	TrCmpAD	TrCmpAU	TrCntLoad	TrCntZero			IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero

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
PWM2INTEN, type R/W, offset 0x0C4, reset 0x0000.0000															
		TrCmpBD	TrCmpBU	TrCmpAD	TrCmpAU	TrCntLoad	TrCntZero			IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM3INTEN, type R/W, offset 0x104, reset 0x0000.0000															
		TrCmpBD	TrCmpBU	TrCmpAD	TrCmpAU	TrCntLoad	TrCntZero			IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM0RIS, type RO, offset 0x048, reset 0x0000.0000															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM1RIS, type RO, offset 0x088, reset 0x0000.0000															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM2RIS, type RO, offset 0x0C8, reset 0x0000.0000															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM3RIS, type RO, offset 0x108, reset 0x0000.0000															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM0ISC, type R/W1C, offset 0x04C, reset 0x0000.0000															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM1ISC, type R/W1C, offset 0x08C, reset 0x0000.0000															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM2ISC, type R/W1C, offset 0x0CC, reset 0x0000.0000															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM3ISC, type R/W1C, offset 0x10C, reset 0x0000.0000															
										IntCmpBD	IntCmpBU	IntCmpAD	IntCmpAU	IntCntLoad	IntCntZero
PWM0LOAD, type R/W, offset 0x050, reset 0x0000.0000															
Load															
PWM1LOAD, type R/W, offset 0x090, reset 0x0000.0000															
Load															
PWM2LOAD, type R/W, offset 0x0D0, reset 0x0000.0000															
Load															
PWM3LOAD, type R/W, offset 0x110, reset 0x0000.0000															
Load															
PWM0COUNT, type RO, offset 0x054, reset 0x0000.0000															
Count															
PWM1COUNT, type RO, offset 0x094, reset 0x0000.0000															
Count															
PWM2COUNT, type RO, offset 0x0D4, reset 0x0000.0000															
Count															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PWM3COUNT, type RO, offset 0x114, reset 0x0000.0000															
Count															
PWM0CMPA, type R/W, offset 0x058, reset 0x0000.0000															
CompA															
PWM1CMPA, type R/W, offset 0x098, reset 0x0000.0000															
CompA															
PWM2CMPA, type R/W, offset 0x0D8, reset 0x0000.0000															
CompA															
PWM3CMPA, type R/W, offset 0x118, reset 0x0000.0000															
CompA															
PWM0CMPB, type R/W, offset 0x05C, reset 0x0000.0000															
CompB															
PWM1CMPB, type R/W, offset 0x09C, reset 0x0000.0000															
CompB															
PWM2CMPB, type R/W, offset 0x0DC, reset 0x0000.0000															
CompB															
PWM3CMPB, type R/W, offset 0x11C, reset 0x0000.0000															
CompB															
PWM0GENA, type R/W, offset 0x060, reset 0x0000.0000															
				ActCmpBD		ActCmpBU		ActCmpAD		ActCmpAU		ActLoad		ActZero	
PWM1GENA, type R/W, offset 0x0A0, reset 0x0000.0000															
				ActCmpBD		ActCmpBU		ActCmpAD		ActCmpAU		ActLoad		ActZero	
PWM2GENA, type R/W, offset 0x0E0, reset 0x0000.0000															
				ActCmpBD		ActCmpBU		ActCmpAD		ActCmpAU		ActLoad		ActZero	
PWM3GENA, type R/W, offset 0x120, reset 0x0000.0000															
				ActCmpBD		ActCmpBU		ActCmpAD		ActCmpAU		ActLoad		ActZero	
PWM0GENB, type R/W, offset 0x064, reset 0x0000.0000															
				ActCmpBD		ActCmpBU		ActCmpAD		ActCmpAU		ActLoad		ActZero	
PWM1GENB, type R/W, offset 0x0A4, reset 0x0000.0000															
				ActCmpBD		ActCmpBU		ActCmpAD		ActCmpAU		ActLoad		ActZero	
PWM2GENB, type R/W, offset 0x0E4, reset 0x0000.0000															
				ActCmpBD		ActCmpBU		ActCmpAD		ActCmpAU		ActLoad		ActZero	
PWM3GENB, type R/W, offset 0x124, reset 0x0000.0000															
				ActCmpBD		ActCmpBU		ActCmpAD		ActCmpAU		ActLoad		ActZero	

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
PWM0DBCTL, type R/W, offset 0x068, reset 0x0000.0000															
															Enable
PWM1DBCTL, type R/W, offset 0x0A8, reset 0x0000.0000															
															Enable
PWM2DBCTL, type R/W, offset 0x0E8, reset 0x0000.0000															
															Enable
PWM3DBCTL, type R/W, offset 0x128, reset 0x0000.0000															
															Enable
PWM0DBRISE, type R/W, offset 0x06C, reset 0x0000.0000															
				RiseDelay											
PWM1DBRISE, type R/W, offset 0x0AC, reset 0x0000.0000															
				RiseDelay											
PWM2DBRISE, type R/W, offset 0x0EC, reset 0x0000.0000															
				RiseDelay											
PWM3DBRISE, type R/W, offset 0x12C, reset 0x0000.0000															
				RiseDelay											
PWM0DBFALL, type R/W, offset 0x070, reset 0x0000.0000															
				FallDelay											
PWM1DBFALL, type R/W, offset 0x0B0, reset 0x0000.0000															
				FallDelay											
PWM2DBFALL, type R/W, offset 0x0F0, reset 0x0000.0000															
				FallDelay											
PWM3DBFALL, type R/W, offset 0x130, reset 0x0000.0000															
				FallDelay											
PWM0FLTSRC0, type R/W, offset 0x074, reset 0x0000.0000															
												FAULT3	FAULT2	FAULT1	FAULT0
PWM1FLTSRC0, type R/W, offset 0x0B4, reset 0x0000.0000															
												FAULT3	FAULT2	FAULT1	FAULT0
PWM2FLTSRC0, type R/W, offset 0x0F4, reset 0x0000.0000															
												FAULT3	FAULT2	FAULT1	FAULT0
PWM3FLTSRC0, type R/W, offset 0x134, reset 0x0000.0000															
												FAULT3	FAULT2	FAULT1	FAULT0
PWM0MINFLTPER, type R/W, offset 0x07C, reset 0x0000.0000															
MFP															

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
PWM1MINFLTPER, type R/W, offset 0x0BC, reset 0x0000.0000															
MFP															
PWM2MINFLTPER, type R/W, offset 0x0FC, reset 0x0000.0000															
MFP															
PWM3MINFLTPER, type R/W, offset 0x13C, reset 0x0000.0000															
MFP															
PWM0FLTSEN, type R/W, offset 0x800, reset 0x0000.0000															
												FAULT3	FAULT2	FAULT1	FAULT0
PWM1FLTSEN, type R/W, offset 0x880, reset 0x0000.0000															
												FAULT3	FAULT2	FAULT1	FAULT0
PWM2FLTSEN, type R/W, offset 0x900, reset 0x0000.0000															
												FAULT3	FAULT2	FAULT1	FAULT0
PWM3FLTSEN, type R/W, offset 0x980, reset 0x0000.0000															
												FAULT3	FAULT2	FAULT1	FAULT0
PWM0FLTSTAT0, type -, offset 0x804, reset 0x0000.0000															
												FAULT3	FAULT2	FAULT1	FAULT0
PWM1FLTSTAT0, type -, offset 0x884, reset 0x0000.0000															
												FAULT3	FAULT2	FAULT1	FAULT0
PWM2FLTSTAT0, type -, offset 0x904, reset 0x0000.0000															
												FAULT3	FAULT2	FAULT1	FAULT0
PWM3FLTSTAT0, type -, offset 0x984, reset 0x0000.0000															
												FAULT3	FAULT2	FAULT1	FAULT0
Quadrature Encoder Interface (QEI) QEI0 base: 0x4002.C000															
QEICTL, type R/W, offset 0x000, reset 0x0000.0000															
			STALLEN	INVI	INVB	INVA	VelDiv			VelEn	ResMode	CapMode	SigMode	Swap	Enable
QEISTAT, type RO, offset 0x004, reset 0x0000.0000															
														Direction	Error
QEIPOS, type R/W, offset 0x008, reset 0x0000.0000															
Position															
Position															
QEIMAXPOS, type R/W, offset 0x00C, reset 0x0000.0000															
MaxPos															
MaxPos															
QEILOAD, type R/W, offset 0x010, reset 0x0000.0000															
Load															
Load															
QEITIME, type RO, offset 0x014, reset 0x0000.0000															
Time															
Time															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QEICOUNT, type RO, offset 0x018, reset 0x0000.0000															
Count															
Count															
QEISPEED, type RO, offset 0x01C, reset 0x0000.0000															
Speed															
Speed															
QEIINTEN, type R/W, offset 0x020, reset 0x0000.0000															
												IntError	IntDir	IntTimer	IntIndex
QEIRIS, type RO, offset 0x024, reset 0x0000.0000															
												IntError	IntDir	IntTimer	IntIndex
QEIISC, type R/W1C, offset 0x028, reset 0x0000.0000															
												IntError	IntDir	IntTimer	IntIndex

D Ordering and Contact Information

D.1 Ordering Information

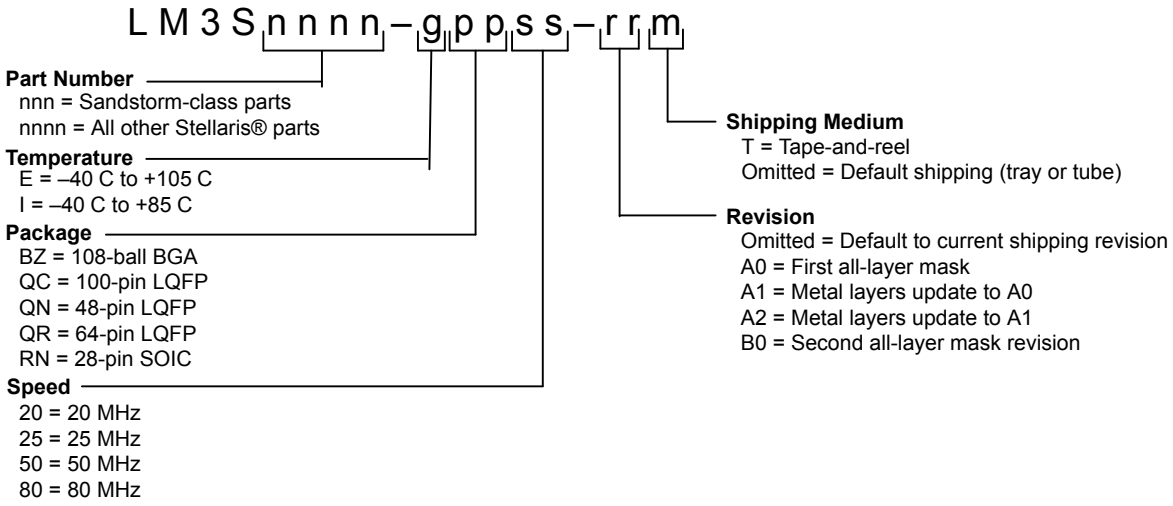


Table D-1. Part Ordering Information

Orderable Part Number	Description
LM3S5749-IQC50	Stellaris® LM3S5749 Microcontroller
LM3S5749-IQC50(T)	Stellaris® LM3S5749 Microcontroller

D.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/kits.html

- Development Kits provide you with all the tools you need to develop and prototype embedded applications right out of the box:

  http://www.luminarymicro.com/products/development_kits.html

See the Luminary Micro website for the latest tools available, or ask your Luminary Micro distributor.

D.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

D.4 Support Information

For support on Luminary Micro products, contact:

support@luminarymicro.com +1-512-279-8800, ext. 3