LUMINARY MICRO™

LM3S101 Microcontroller

DATA SHEET

DS-LM3S101-00

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.
2499 South Capital of Texas Hwy, Suite A-100
Austin, TX 78746
Main: +1-512-279-8800
Fax: +1-512-279-8879
http://www.luminarymicro.com

Table of Contents

List of Figures

List of Tables

List of Registers

Revision History

This table provides a summary of the document revisions.

Date	Revision	Description
March 2006	00	Initial public release.

About This Document

This data sheet provides reference information for the LM3S101 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:

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

This documentation list was current as of publication date. Please check our web site at www.luminarymicro.com for additional related documentation, including application notes and white papers.

Documentation Conventions

This document uses the conventions shown in Table 0-1.

Table 0-1. Documentation Conventions

Notation	Meaning
General Register Notation	
REGISTER	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 Table 3-1, "Memory Map," on page 29.
Register *N*	Registers are numbered consecutively throughout the document to aid in referencing them. The register number has no meaning to software.

Table 0-1. Documentation Conventions

Notation	Meaning
reserved	Register bits marked reserved are reserved for future use. Reserved bits return an indeterminate value, and should never be changed. Only write a reserved bit with its current value.
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.
RO	Software can read this field. Always write the chip reset value.
R/W	Software can read or write this field.
R/W1C	Software can read or write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged. This register type is primarily used for clearing interrupt status bits where the read operation provides the interrupt status and the write of the read value clears only the interrupts being reported at the time the register was read.
W1C	Software can write this field. A write of a 0 to a W1C bit does not affect the bit value in the register. A write of a 1 clears the value of the bit in the register; the remaining bits remain unchanged. A read of the register returns no meaningful data. This register is typically used to clear the corresponding bit in an interrupt register.
WO	Only a write by software is valid; a read of the register returns no meaningful data.
Register Bit/Field Reset Value	This value in the register bit diagram shows the bit/field value after any reset, unless noted.
0	Bit cleared to 0 on chip reset.
1	Bit set to 1 on chip reset.
–	Nondeterministic.
Pin/Signal Notation	
[ ]	Pin alternate function; a pin defaults to the signal without the brackets.
pin	Refers to the physical connection on the package.
signal	Refers to the electrical signal encoding of a pin.
assert a signal	Change the value of the signal from the logically False state to the logically True state. For active High signals, the asserted signal value is 1 (High); for active Low signals, the asserted signal value is 0 (Low). The active polarity (High or Low) is defined by the signal name (see SIGNAL and $\overline{\text{SIGNAL}}$ below).

Table 0-1. Documentation Conventions

Notation	Meaning
deassert a signal	Change the value of the signal from the logically True state to the logically False state.
$\overline{\texttt{SIGNAL}}$	Signal names are in uppercase and in the Courier font. An overbar on a signal name indicates that it is active Low. To assert $\overline{\texttt{SIGNAL}}$ is to drive it Low; to deassert $\overline{\texttt{SIGNAL}}$ is to drive it High.
SIGNAL	Signal names are in uppercase and in the Courier font. An active High signal has no overbar. To assert SIGNAL is to drive it High; to deassert SIGNAL is to drive it Low.
Numbers	
X	An uppercase X indicates any of several values is allowed, where X can be any legal pattern. For example, a binary value of 0X00 can be either 0100 or 0000, a hex value of 0xX is 0x0 or 0x1, and so on.
0x	Hexadecimal numbers have a prefix of 0x. For example, 0x00FF is the hexadecimal number FF. Binary numbers are indicated with a b suffix, for example, 1011b. Decimal numbers are written without a prefix or suffix.

1 Architectural Overview

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

The LM3S101 controller in the Stellaris family offers the advantages of ARM's widely available development tools, System-on-Chip (SoC) infrastructure IP applications, and a large user community. Additionally, the controller uses ARM's Thumb®-compatible Thumb-2 instruction set to reduce memory requirements and, thereby, cost.

Luminary Micro offers a complete solution to get to market quickly, with a customer development board, white papers and application notes, and a strong support, sales, and distributor network.

1.1 Product Features

The LM3S101 microcontroller includes the following product features:

- 32-Bit RISC Performance
  - 32-bit ARM® Cortex™-M3 v7M architecture optimized for small-footprint embedded applications
  - Thumb®-compatible Thumb-2-only instruction set processor core for high code density
  - 20-MHz operation
  - Hardware-division and single-cycle-multiplication
  - Integrated Nested Vectored Interrupt Controller (NVIC) providing deterministic interrupt handling
  - 14 interrupts with eight priority levels
  - Unaligned data access, enabling data to be efficiently packed into memory
  - Atomic bit manipulation (bit-banding) delivers maximum memory utilization and streamlined peripheral control
- Internal Memory
  - 8 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
  - 2 KB single-cycle SRAM
- General-Purpose Timers
  - Two timers, each of which can be configured as a single 32-bit timer or as two 16-bit timers
  - 32-bit Timer modes:
    - Programmable one-shot timer
    - Programmable periodic timer
    - Real-Time Clock when using an external 32-KHz clock as the input
    - User-enabled stalling in periodic and one-shot mode when the controller asserts the CPU Halt flag during debug
  - 16-bit Timer modes:

    - General-purpose timer function with an 8-bit prescaler
    - Programmable one-shot timer
    - Programmable periodic timer
    - User-enabled stalling when the controller asserts CPU Halt flag during debug
  - 16-bit Input Capture modes:
    - Input edge count capture
    - Input edge time capture
  - 16-bit PWM mode:
    - Simple PWM mode with software-programmable output inversion of the PWM signal
- ARM FiRM-compliant Watchdog Timer
  - 32-bit down counter with a programmable load register
  - Separate watchdog clock with an enable
  - Programmable interrupt generation logic with interrupt masking
  - Lock register protection from runaway software
  - Reset generation logic with an enable/disable
  - ARM PrimeCell®-compliant peripheral and cell identification registers
  - User-enabled stalling when the controller asserts the CPU Halt flag during debug
- Synchronous Serial Interface (SSI)
  - Master or slave operation
  - Programmable clock bit rate and prescale
  - Separate transmit and receive FIFOs, 16 bits wide, 8 locations deep
  - Programmable interface operation for Freescale SPI, National Semiconductor MICROWIRE™, or Texas Instruments synchronous serial interfaces
  - Programmable data frame size from 4 to 16 bits
  - Internal loopback test mode for diagnostic/debug testing
- UART
  - Fully programmable 16C550-type UART
  - Separate 16x8 transmit (TX) and 16x12 receive (RX) FIFOs to reduce CPU interrupt service loading
  - Programmable baud-rate generator with fractional divider
  - Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
  - FIFO trigger levels of 1/8, 1/4, 1/2, 3/4 and 7/8
  - Standard asynchronous communication bits for start, stop and parity
  - False start bit detection
  - Line-break generation and detection
- Analog Comparators
  - Two independent integrated analog comparators

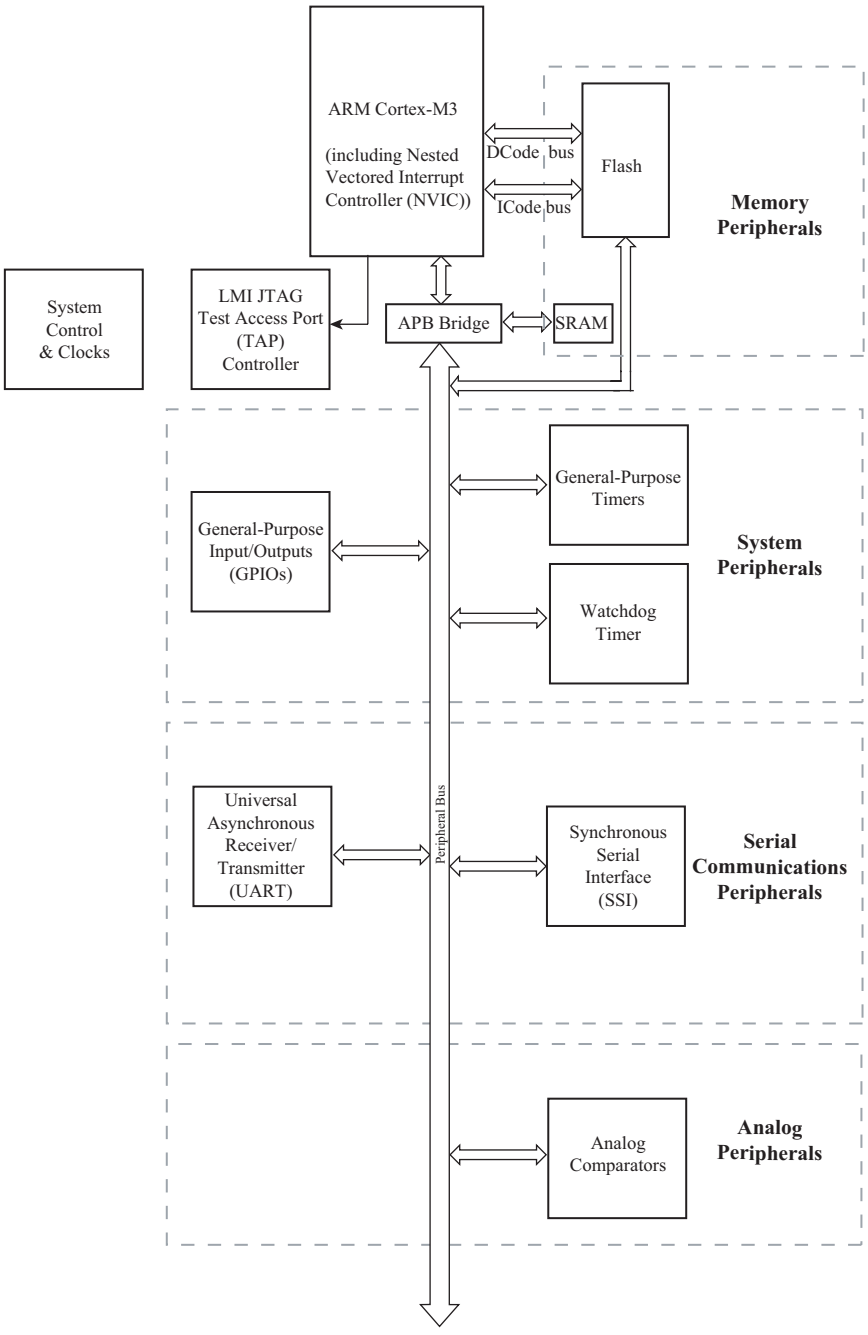
  - Configurable for output to drive an output pin or generate an interrupt
  - Compare external pin input to external pin input or to internal programmable voltage reference
- GPIOs
  - 2 to 18 GPIOs, depending on configuration
  - Programmable interrupt generation as either edge-triggered or level-sensitive
  - Bit masking in both read and write operations through address lines
  - Programmable control for GPIO pad configuration:
    - Weak pull-up or pull-down resistors
    - 2-mA, 4-mA, and 8-mA pad drive
    - Slew rate control for the 8-mA drive
    - Open drain enables
    - Digital input enables
- Power
  - On-chip Linear Drop-Out (LDO) voltage regulator, with programmable output user-adjustable from 2.25 V to 2.75 V
  - Low-power options on controller: Sleep and Deep-sleep modes
  - Low-power options for peripherals: software controls shutdown of individual peripherals
  - User-enabled LDO unregulated voltage detection and automatic reset
  - 3.3-V supply brownout 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 linear drop-out (LDO) regulator output goes unregulated
- Additional Features
  - Six reset sources
  - Programmable clock source control
  - Clock gating to individual peripherals for power savings
  - IEEE 1149.1-1990 compliant Test Access Port (TAP) controller
  - Debug access via JTAG and Serial Wire interfaces
  - Full JTAG boundary scan
- Package
  - 28-pin RoHS-compliant SOIC
  - Commercial and industrial operating temperatures

1.2 Target Applications

- Factory automation and control
- Industrial control power devices
- Building and home automation

1.3 High-Level Block Diagram

Figure 1-1. Stellaris High-Level Block Diagram



1.4 Functional Overview

The following sections provide an overview of the features of the LM3S101 microcontroller. The chapter number in parenthesis indicates where that feature is discussed in detail. Ordering and support information can be found in “Contact Information” on page 506.

1.4.1 ARM Cortex™-M3

1.4.1.1 Processor Core (Section 2 on page 26)

All members of the Stellaris product family, including the LM3S101 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.

Section 2, “ARM Cortex-M3 Processor Core,” on page 26 provides an overview of the ARM core; the core is detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

1.4.1.2 Nested Vectored Interrupt Controller (NVIC)

The LM3S101 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 seven exceptions (system handlers) and 14 interrupts.

Section 4, “Interrupts,” on page 31 provides an overview of the NVIC controller and the interrupt map. Exceptions and interrupts are detailed in the *ARM® Cortex™-M3 Technical Reference Manual*.

1.4.2 Motor Control Peripherals

To enhance motor control, the LM3S101 controller features Pulse Width Modulation (PWM) outputs.

1.4.2.1 PWM (“16-Bit PWM Mode” on page 139)

Pulse Width Modulation (PWM) is a powerful technique often used to regulate a voltage by holding the frequency constant and varying the pulse width.

On the LM3S101, PWM motion control functionality can be achieved through the motion control features of the general-purpose timers (using the CCP pins).

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

1.4.3 Analog Peripherals

To handle analog signals, the LM3S101 controller offers two analog comparators.

1.4.3.1 Analog Comparators (Section 13 on page 251)

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

The LM3S101 controller provides two independent integrated analog comparators that can be configured to drive an output or generate an interrupt.

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

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

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

1.4.4 Serial Communications Peripherals

The LM3S101 controller supports both asynchronous and synchronous serial communications with one fully programmable 16C550-type UART and SSI serial communications.

1.4.4.1 UART (Section 11 on page 182)

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 LM3S101 controller includes one fully programmable 16C550-type UART that supports data transfer speeds up to 460.8 Kbps. (Although similar in functionality to a 16C550 UART, it is not register compatible.)

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

1.4.4.2 SSI (Section 12 on page 218)

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

The LM3S101 controller SSI module provides the functionality for synchronous serial communications with peripheral devices, and can be configured to use the Freescale SPI, National Semiconductor MICROWIRE, or TI synchronous serial interface frame formats. The size of the data frame is also configurable, and can be set to be between 4 and 16 bits, inclusive.

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

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

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

The Inter-Integrated Circuit (I^2C) bus provides bi-directional data transfer through a two-wire design (a serial data line SDL 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 Stellaris I^2C module provides the ability to communicate to other IC devices over an I^2C bus. The I^2C bus supports devices that can both transmit and receive (write and read) data.

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

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

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

1.4.5 System Peripherals

1.4.5.1 Programmable GPIOs (Section 8 on page 93)

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

The LM3S101 controller GPIO module is composed of three 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 2 to 18 programmable input/output pins. The number of GPIOs available depends on the peripherals being used (see Table 15-4 on page 268 for the signals available to each GPIO pin).

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

1.4.5.2 Two Programmable Timers (Section 9 on page 130)

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

The LM3S101 controller General-Purpose Timer Module (GPTM) contains two GPTM blocks. Each GPTM block provides two 16-bit timer/counters that can be configured to operate independently as timers or event counters, or configured to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC).

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

1.4.5.3 Watchdog Timer (Section 10 on page 160)

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 LM3S101 controller 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 LM3S101 controller offers both SRAM and Flash memory.

1.4.6.1 SRAM (Section 7.2.1 on page 80)

The LM3S101 static random access memory (SRAM) controller supports 2 KB SRAM. The internal SRAM of the Stellaris devices is located at address 0x20000000 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 (Section 7.2.2 on page 81)

The LM3S101 Flash controller supports 8 KB of flash memory. The flash is organized as a set of 1-KB blocks that can be individually erased. Erasing a block causes the entire contents of the block to be reset to all 1s. These blocks are paired into a set of 2-KB blocks that can be individually protected. The blocks can be marked as read-only or execute-only, providing different levels of code protection. Read-only blocks cannot be erased or programmed, protecting the contents of those blocks from being modified. Execute-only blocks cannot be erased or programmed, and can only be read by the controller instruction fetch mechanism, protecting the contents of those blocks from being read by either the controller or by a debugger.

1.4.7 Additional Features

1.4.7.1 Memory Map (Section 3 on page 29)

A memory map lists the location of instructions and data in memory. The memory map for the LM3S101 controller can be found on page 29. 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 (Section 5 on page 34)

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

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

The LMI 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 LMI JTAG instructions select the LMI TDO outputs. The multiplexer is controlled by the LMI JTAG controller, which has comprehensive programming for the ARM, LMI, and unimplemented JTAG instructions.

1.4.7.3 System Control and Clocks (Section 6 on page 44)

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

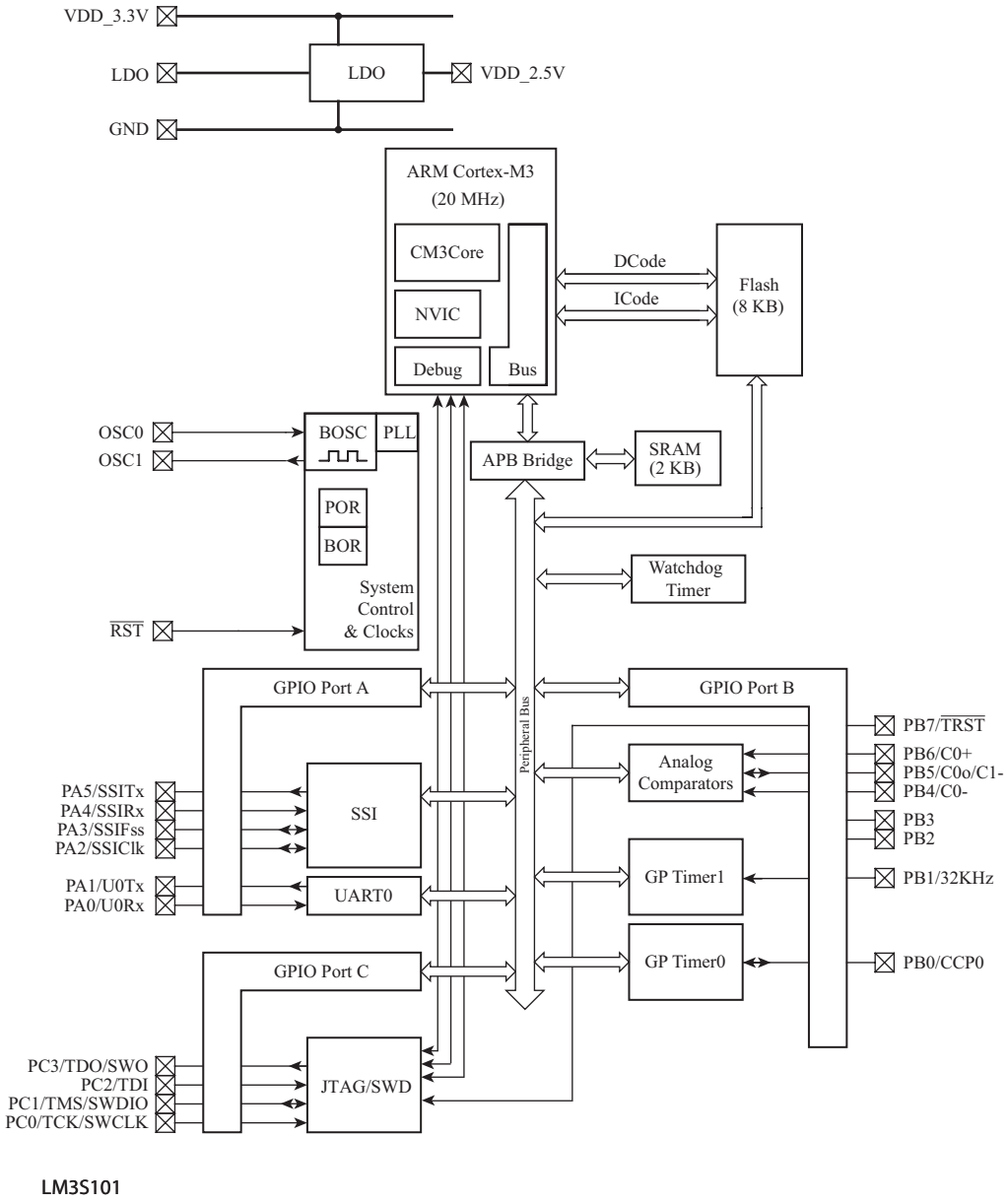
1.4.8 Hardware Details

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

- Section 14, “Pin Diagram,” on page 262
- Section 15, “Signal Tables,” on page 263
- Section 16, “Operating Characteristics,” on page 270
- Section 17, “Electrical Characteristics,” on page 271
- Section 18, “Package Information,” on page 282

1.5 System Block Diagram

Figure 1-2. Stellaris System-Level Block Diagram



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:

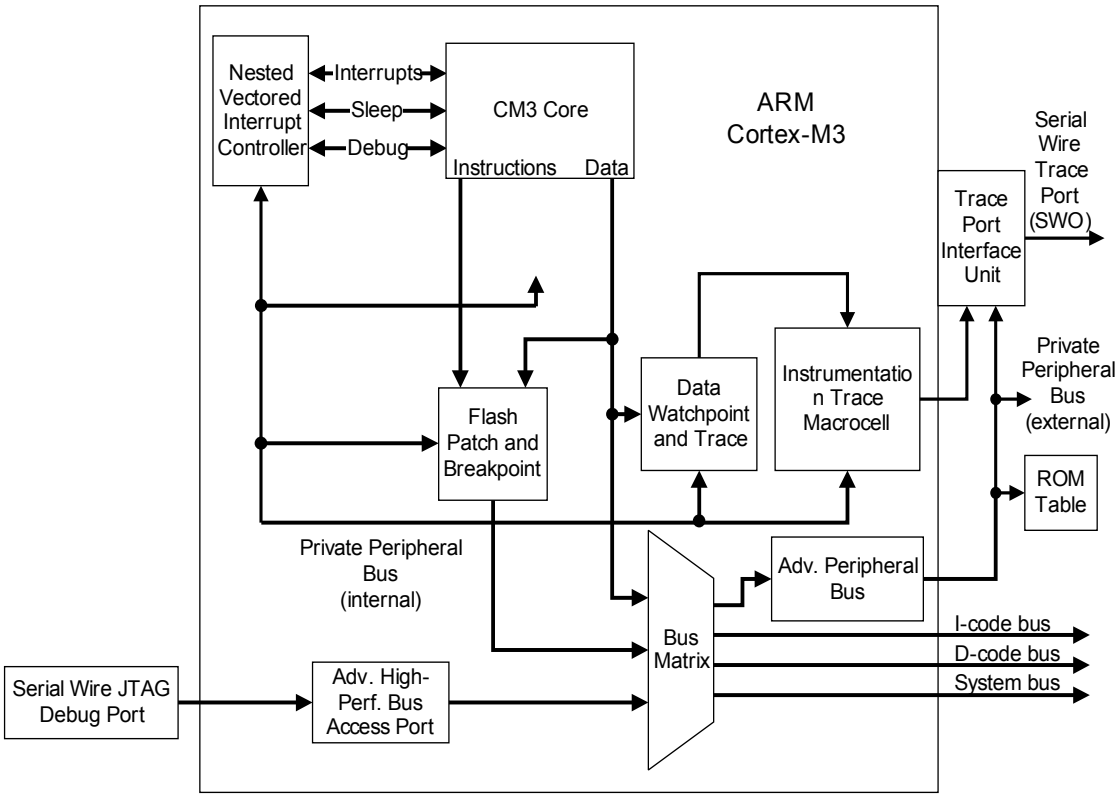
- Compact core.
- Thumb-2 instruction set, delivering the high-performance expected of an ARM core in the memory size usually associated with 8- and 16-bit devices; typically in the range of a few kilobytes of memory for microcontroller class applications.
- Exceptional interrupt handling, by implementing the register manipulations required for handling an interrupt in hardware.
- Full-featured debug solution with a:
  - Serial Wire JTAG Debug Port (SWJ-DP)
  - Flash Patch and Breakpoint (FPB) unit for implementing breakpoints
  - Data Watchpoint and Trigger (DWT) unit for implementing watchpoints, trigger resources, and system profiling
  - Instrumentation Trace Macrocell (ITM) for support of printf style debugging
  - Trace Port Interface Unit (TPIU) for bridging to a Trace Port Analyzer

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

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 *CoreSight™ Design Kit Technical Reference Manual*.

2.1 Block Diagram

Figure 2-1. CPU High-Level 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. 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).

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 the 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.

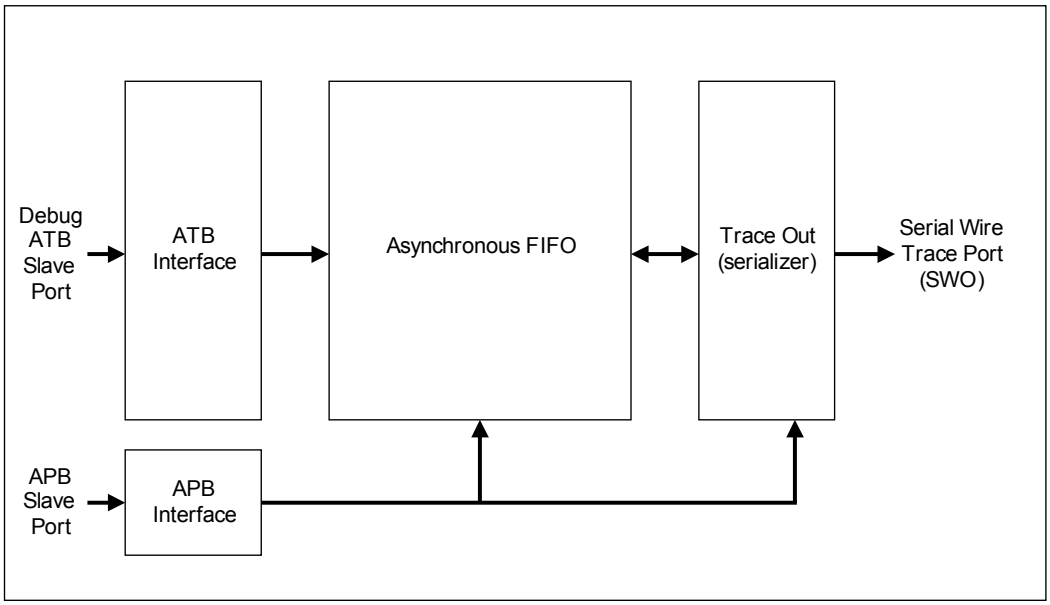
2.2.2 Embedded Trace Macrocell (ETM)

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. 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 LM3S101 controller does not include the memory protection unit (MPU) of the ARM Cortex-M3.

2.2.6 Nested Vectored Interrupt Controller (NVIC)

2.2.6.1 Interrupts

The *ARM® Cortex™-M3 Technical Reference Manual* describes the maximum number of interrupts and interrupt priorities. The Stellaris microcontrollers support 14 interrupts with eight priority levels.

2.2.6.2 SysTick Calibration Value Registers

The SysTick Calibration Value register is not implemented.

3 Memory Map

The memory map for the LM3S101 is provided in Table 3-1. In this manual, register addresses are given as a hexadecimal increment, relative to the module's base address as shown in the memory map. See also Chapter 4, "Memory Map" in the *ARM® Cortex™-M3 Technical Reference Manual*.

Table 3-1. Memory Map (Sheet 1 of 2)

Start	End	Description	For details on registers, see ...
Memory			
0x00000000	0x1FFFFFFF	On-chip flash[a]	page 83
0x20000000	0x200FFFFF	Bit-banded on-chip SRAM[b]	-
0x20100000	0x21FFFFFF	Reserved non-bit banded SRAM space[c]	-
0x22000000	0x23FFFFFF	Bit-band alias of 0x20000000 through 0x200FFFFF	-
0x24000000	0x3FFFFFFF	Reserved non-bit-banded SRAM space	-
FiRM Peripherals			
0x40000000	0x40000FFF	Watchdog timer	page 162
0x40001000	0x40003FFF	Reserved for three additional watchdog timers (per FiRM specification)	-
0x40004000	0x40004FFF	GPIO Port A	page 99
0x40005000	0x40005FFF	GPIO Port B	page 99
0x40006000	0x40006FFF	GPIO Port C	page 99
0x40007000	0x40007FFF	Reserved for additional GPIO port (per FiRM specification)	-
0x40008000	0x40008FFF	SSI	page 228
0x40009000	0x4000BFFF	Reserved for three additional SSIs (per FiRM specification)	-
0x4000C000	0x4000CFFF	UART0	page 188
0x4000D000	0x4000FFFF	Reserved for additional UART (per FiRM specification)	-
0x40010000	0x4001FFFF	Reserved for future FiRM peripherals	-
Peripherals			
0x40020000	0x40023FFF	Reserved	-
0x40024000	0x40027FFF	Reserved	-
0x40028000	0x4002BFFF	Reserved	-
0x4002C000	0x4002FFFF	Reserved	-
0x40030000	0x40030FFF	Timer0	page 140

Table 3-1. Memory Map (Sheet 2 of 2)

Start	End	Description	For details on registers, see ...
0x40031000	0x40031FFF	Timer1	page 140
0x40032000	0x40037FFF	Reserved	-
0x40038000	0x4003BFFF	Reserved	-
0x4003C000	0x4003CFFF	Analog comparators	page 254
0x4003D000	0x400FCFFF	Reserved	-
0x400FD000	0x400FDFFF	Flash control	page 83
0x400FE000	0x400FFFFF	System control	page 51
0x40100000	0x41FFFFFF	Reserved	-
0x42000000	0x43FFFFFF	Bit-band alias of 0x40000000 through 0x400FFFFF	-
0x44000000	0xDFFFFFFF	Reserved	-
Private Peripheral Bus			
0xE0000000	0xE0000FFF	Instrumentation Trace Macrocell (ITM)	*ARM® Cortex™-M3 Technical Reference Manual*
0xE0001000	0xE0001FFF	Data Watchpoint and Trace (DWT)	
0xE0002000	0xE0002FFF	Flash Patch and Breakpoint (FPB)	
0xE0003000	0xE000DFFF	Reserved	
0xE000E000	0xE000EFFF	Nested Vectored Interrupt Controller (NVIC)	
0xE000F000	0xE003FFFF	Reserved	
0xE0040000	0xE0040FFF	Trace Port Interface Unit (TPIU)	
0xE0041000	0xE0041FFF	Reserved	-
0xE0042000	0xE00FFFFF	Reserved	-
0xE0100000	0xFFFFFFFF	Reserved for vendor peripherals	-

a. The available flash aliases throughout this address range.
b. The available SRAM aliases throughout this address range.
c. All reserved space returns random results when read and ignores writes.

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 lists all the exceptions. Software can set eight priority levels on seven of these exceptions (system handlers) as well as on 14 interrupts (listed in Table 4-2). Priorities on the system handlers are set with the NVIC **System Handler Priority** registers. Interrupts are enabled through the NVIC **Interrupt Set Enable** register and prioritized with the NVIC **Interrupt Priority** registers. You can also group priorities by splitting priority levels into pre-emption priorities and subpriorities. All the interrupt registers are described in Chapter 8, “Nested Vectored Interrupt Controller” in the *ARM® Cortex™-M3 Technical Reference Manual*.

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

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

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

Table 4-1. Exception Types

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

Table 4-1. Exception Types

Exception Type	Position	Priority[a]	Description
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 lists the interrupts on the LM3S101 controller.

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

Table 4-2. Interrupts

Interrupt (Bit in Interrupt Registers)	Description
0	GPIO Port A
1	GPIO Port B
2	GPIO Port C
3-4	Reserved
5	UART0
6	Reserved
7	SSI
8-17	Reserved
18	Watchdog timer
19	Timer0a
20	Timer0b
21	Timer1a
22	Timer1b

Table 4-2. Interrupts

Interrupt (Bit in Interrupt Registers)	Description
23-24	Reserved
25	Analog Comparator 0
26	Analog Comparator 1
27	Reserved
28	System Control
29	Flash Control
30-31	Reserved

5 JTAG Interface

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

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

The LMI 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 LMI JTAG instructions select the LMI TDO outputs. The multiplexer is controlled by the LMI JTAG controller, which has comprehensive programming for the ARM, LMI, and unimplemented JTAG instructions.

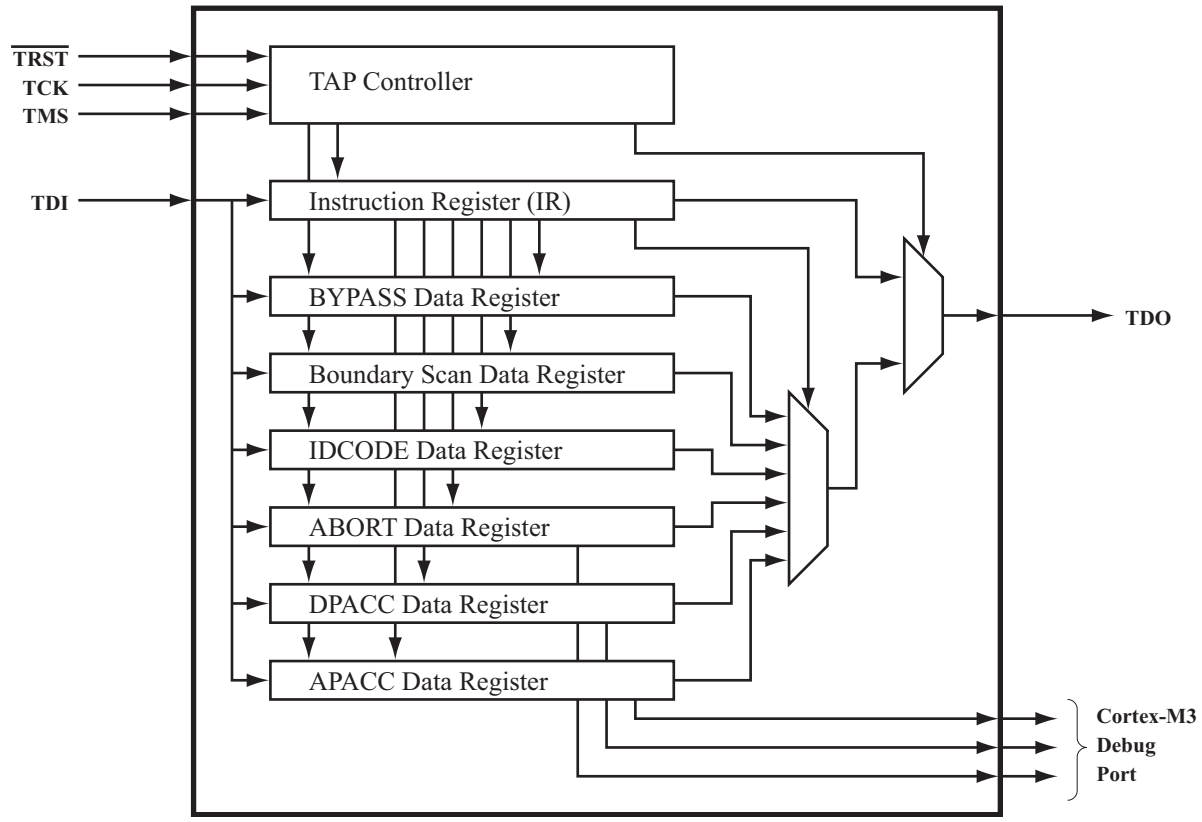
The JTAG module has the following features:

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

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

5.1 Block Diagram

Figure 5-1. JTAG Module Block Diagram



5.2 Functional Description

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

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

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

See “JTAG and Boundary Scan” on page 277 for JTAG timing diagrams.

5.2.1 JTAG Interface Pins

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

Table 5-1. JTAG Port Pins Reset State

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

5.2.1.1 Test Reset Input (TRST)

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

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

5.2.1.2 Test Clock Input (TCK)

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

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

5.2.1.3 Test Mode Select (TMS)

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

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

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

5.2.1.4 Test Data Input (TDI)

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

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

5.2.1.5 Test Data Output (TDO)

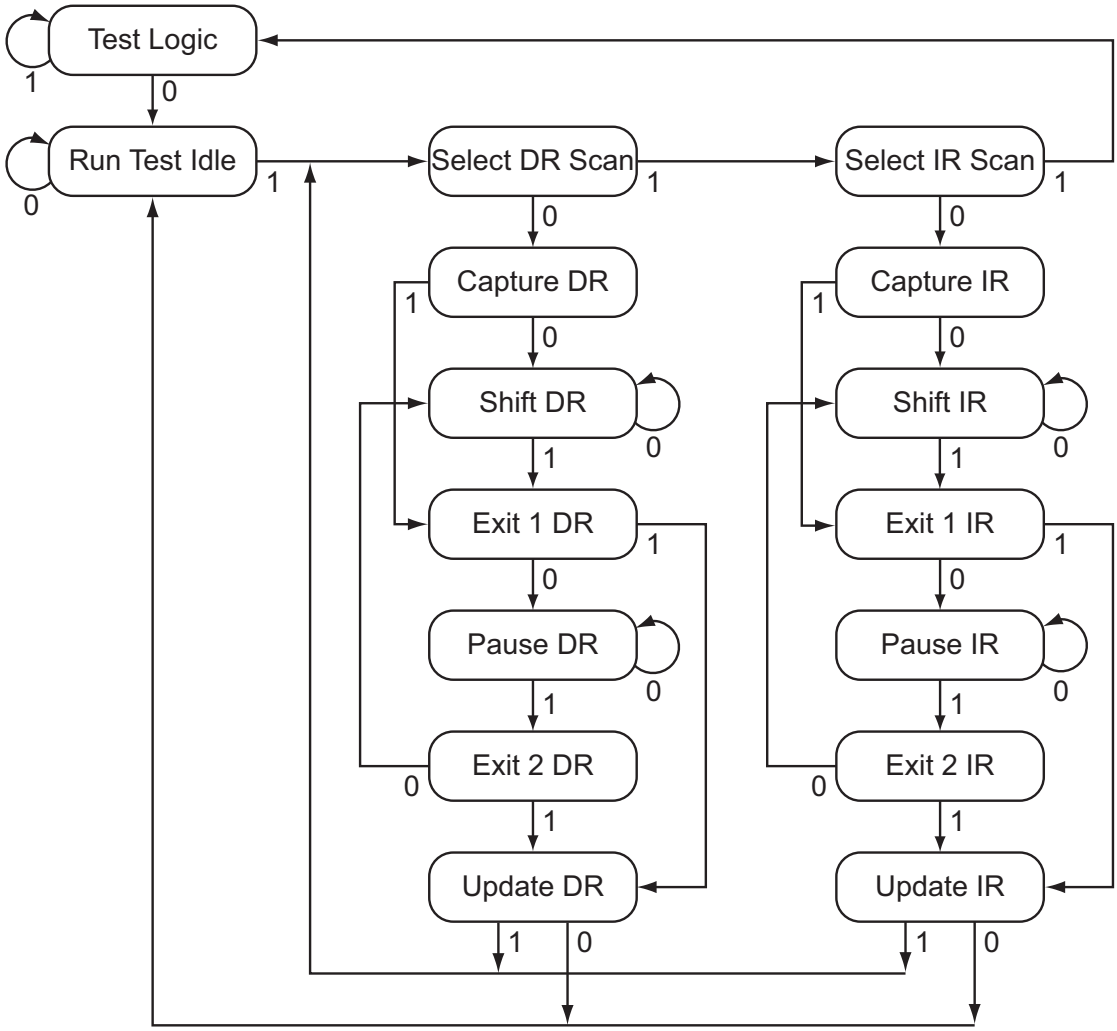
The TDO pin provides an output stream of serial information from the IR chain or the DR chains. The value of TDO depends on the current TAP state, the current instruction, and the data in the chain being accessed. In order to save power when the JTAG port is not being used, the TDO pin is placed in an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

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

5.2.2 JTAG TAP Controller

The JTAG TAP controller state machine is shown in Figure 5-2 on page 38. The TAP controller state machine is reset to the Test-Logic-Reset state on the assertion of a Power-On-Reset (POR) or the assertion of $\overline{\text{TRST}}$. Asserting the correct sequence on the TMS pin allows the JTAG module to shift in new instructions, shift in data, or idle during extended testing sequences. For detailed information on the function of the TAP controller and the operations that occur in each state, please refer to *IEEE Standard 1149.1*.

Figure 5-2. Test Access Port State Machine



5.2.3 Shift Registers

The shift registers consist of a serial shift register chain and a parallel load register. The serial shift register chain samples specific information during the TAP controller's CAPTURE states and allows this information to be shifted out of TDO during the TAP controller's SHIFT states. While the sampled data is being shifted out of the chain on TDO, new data is being shifted into the serial shift register on TDI. This new data is stored in the parallel load register during the TAP controller's UPDATE states. Each of the shift registers is discussed in detail in "Shift Registers" on page 38.

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 requires clarification.

5.2.4.1 GPIO Functionality

Caution – If the JTAG pins will be used as GPIOs, 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 will not have enough time to connect and halt the controller before the JTAG pin functionality switches. This locks the debugger out of the part. This can be avoided with a software routine that restores JTAG functionality using an external trigger.

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

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

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

5.2.4.2 ARM Serial Wire Debug (SWD)

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

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

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

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

5.3 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.

5.3.1 Instruction Register (IR)

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

Table 5-2. JTAG Instruction Register Commands

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

5.3.1.1 EXTEST Instruction

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

5.3.1.2 INTEST Instruction

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

5.3.1.3 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 is 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 42 for more information.

5.3.1.4 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 43 for more information.

5.3.1.5 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 43 for more information.

5.3.1.6 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 43 for more information.

5.3.1.7 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 controller. This information can be used by testing equipment and debuggers to automatically configure their input and output data streams. IDCODE is the default instruction that is loaded into the JTAG Instruction Register when a power-on-reset (POR) is asserted, $\overline{\text{TRST}}$ is asserted, or the Test-Logic-Reset state is entered. Please see “IDCODE Data Register” on page 42 for more information.

5.3.1.8 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 42 for more information.

5.3.2 Data Registers

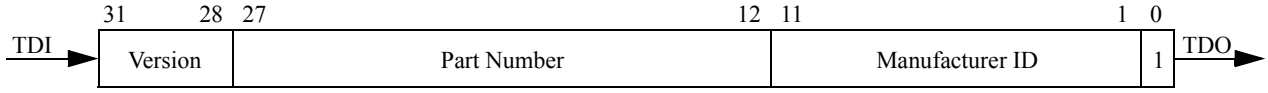
The JTAG module contains six data registers. These include: IDCODE, BYPASS, Boundary Scan, APACC, DPACC, and ABORT serial data register chains. Each of these data registers is discussed in the following sections.

5.3.2.1 IDCODE Data Register

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

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

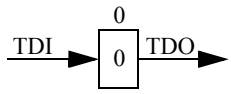
Figure 5-3. IDCODE Register Format



5.3.2.2 BYPASS Data Register

The format for the 1-bit BYPASS data register defined by the *IEEE Standard 1149.1* is shown in Figure 5-4. The standard requires that every JTAG-compliant 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

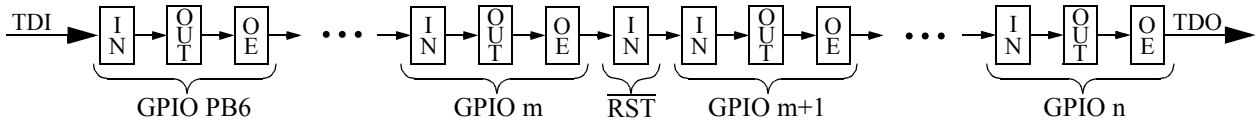


5.3.2.3 Boundary Scan Data Register

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

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

Figure 5-5. Boundary Scan Register Format



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

5.3.2.4 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*.

5.3.2.5 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*.

5.3.2.6 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 of the device and individual peripherals, and handles reset detection and reporting.

6.1 Functional Description

The System Control module provides the following capabilities:

- Device identification, see page 44
- Local control, such as reset (see page 44), power (see page 47) and clock control (see page 47)
- System control (Run, Sleep, and Deep-Sleep modes), see page 49

6.1.1 Device Identification

Seven read-only registers provide software with information on the microcontroller, such as version, part number, SRAM size, Flash size, and other features. See the **DID0**, **DID1** and **DC0**-**DC4** registers starting on page 52.

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 page 44.
2. Power-on reset (POR), see page 45.
3. Internal brown-out (BOR) detector, see page 45.
4. Software-initiated reset (with the Software Reset registers), see page 46.
5. A watchdog timer reset condition violation, see page 46.
6. Internal linear drop-out (LDO) regulator output, see page 47.

After a reset, the **Reset Cause (RESC)** register (see page 69) is set with the reset cause. The bits in this register are sticky and maintain their state across multiple reset sequences, except when an external reset is the cause, and then all the other bits in the **RESC** register are cleared.

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

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

The external reset pin ($\overline{\text{RST}}$) resets the controller. This resets the core and all the peripherals except the JTAG TAP controller (see “JTAG Interface” on page 34). The external reset sequence is as follows:

1. The external reset pin ($\overline{\text{RST}}$) is asserted and then de-asserted.
2. After $\overline{\text{RST}}$ is de-assserted, the main crystal oscillator must be allowed to settle and there is an internal main oscillator counter that takes from 15-30 ms to account for this. During this time, internal reset to the rest of the controller is held active.

3. The internal reset is released and the controller fetches and loads the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

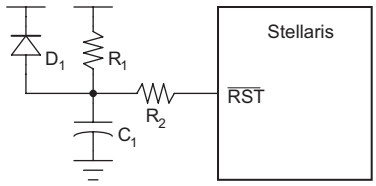
The external reset timing is shown in Figure 17-8 on page 280.

6.1.2.3 Power-On Reset (POR)

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

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

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

The internal POR is only active on the initial power-up of the controller. The Power-On Reset timing is shown in Figure 17-9 on page 280.

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

Brown-out resets are controlled with the **Power-On and Brown-Out Reset Control (PBORCTL)** register (see page 60). The BORIOR bit in the **PBORCTL** register must be set for a brown-out to trigger a reset. The brown-out reset sequence is as follows:

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

The internal Brown-Out Reset timing is shown in Figure 17-10 on page 280.

6.1.2.5 Software Reset

Each peripheral can be reset by software. There are three registers that control this function (see the **SRCRn** registers, starting on page 62). If the bit position corresponding to a peripheral is set, the peripheral is reset. The encoding of the reset registers is consistent with the encoding of the clock gating control for peripherals and on-chip functions (see “System Control” on page 49). Writing a bit lane with a value of 1 initiates a reset of the corresponding unit. Note that all reset signals for all clocks of the specified unit are asserted as a result of a software-initiated reset.

The entire system can be reset by software also. 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 in 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 released and the controller fetches and loads 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 17-11 on page 281.

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 (see page 163), 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 fetches and loads the initial stack pointer, the initial program counter, and the first instruction designated by the program counter, and then begins execution.

The watchdog reset timing is shown in Figure 17-12 on page 281.

6.1.2.7 Linear Drop-Out

A reset can be made when the internal linear drop-out (LDO) regulator output goes unregulated. This is initially disabled and may be enabled by software. LDO is controlled with the **LDO Power Control (LDOPCTL)** register (see page 61). The LDO reset sequence is as follows:

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

The LDO reset timing is shown in Figure 17-13 on page 281.

6.1.3 Power Control

The LDO regulator permits the adjustment of the on-chip output voltage (V_{OUT}). The output may be adjusted in 50 mV increments between the range of 2.25 V through 2.75 V. The adjustment is made through the `VADJ` field of the **LDO Power Control (LDOPCTL)** register (see page 61).

6.1.4 Clock Control

System control determines the clocking and control of clocks in this part.

6.1.4.1 Fundamental Clock Sources

There are two fundamental clock sources for use in the device:

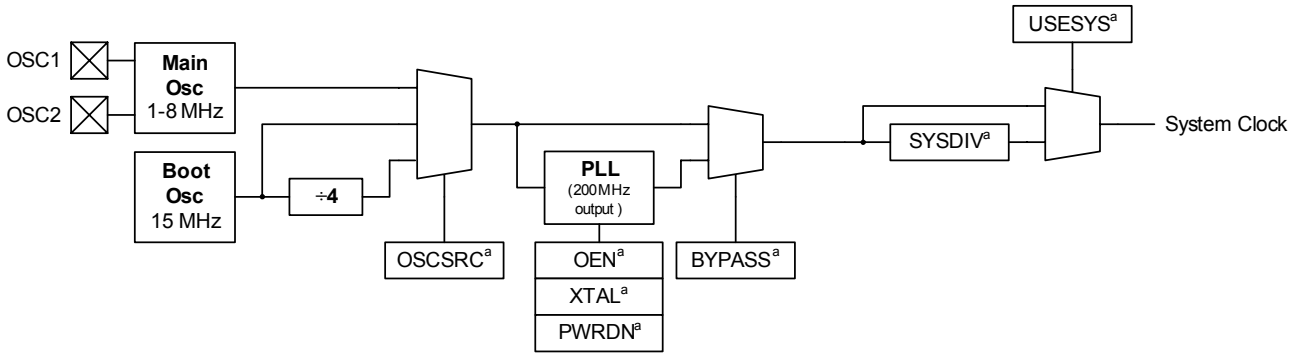
- The main oscillator, driven from either an external crystal or a single-ended source. As a crystal, the main oscillator source is specified to run from 1-8 MHz. However, when the crystal is being used as the PLL source, it must be from 5-8 MHz to meet PLL requirements. As a single-ended source, the range is from DC to the specified speed of the device.
- The boot oscillator, which is an on-chip free running clock. The boot oscillator is specified to run at 15 MHz ± 30%. It can be used to clock the system but the tolerance of frequency range must be met.

The internal system clock may be driven by either of the above two reference sources as well as the internal PLL, provided that the PLL input is connected to a clock source that meets its AC requirements.

Nearly all of the control for the clocks is provided by the **Run-Mode Clock Configuration (RCC)** register (see page 70).

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

Figure 6-2. Main Clock Tree



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

6.1.4.2 PLL Frequency Configuration

The user does not have direct control over the PLL frequency, but is required to match the external crystal used to an internal PLL-Crystal table. This table is used to create the best fit for PLL parameters to the crystal chosen. Not all crystals result in the PLL operating at exactly 200 MHz, though the frequency will be within ±1%; non-exact values are fine, if tolerated by the system. The result of the lookup is kept in the **XTAL to PLL Translation (PLLCTL)** register (see page 74).

Table 6-3 on page 72 describes the available crystal choices and default programming of the **PLLCTL** register. The crystal number is written into the XTAL field of the **Run-Mode Clock Configuration (RCC)** register (see page 70). Any time the XTAL field changes, a read of the internal table is performed to get the correct value. Table 6-3 on page 72 describes the available crystal choices and default programming values.

6.1.4.3 PLL Modes

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

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

The modes are programmed using the **Run-Mode Clock Configuration (RCC)** register fields as shown in Table 6-4 on page 73.

6.1.4.4 PLL Operation

If the PLL configuration is changed, the PLL output is not stable for a period of time (PLL T_{READY}=0.5 ms) and during this time, the PLL is not usable as a clock reference.

The PLL is changed by one of the following:

- Change to the XTAL value in the **Run-Mode Clock Configuration (RCC)** register (see page 70)—writes of the same value will not cause a relock).
- Change in the PLL from Power-Down to Normal mode.

A counter is defined to measure the T_{READY} requirement. The counter is clocked by the boot oscillator. The range of the boot oscillator has been taken into account and the down counter is set to 0x3000 (that is, ~800 us at a 15-MHz boot oscillator clock). Hardware is provided to keep the PLL from being used as a system clock until the T_{READY} condition is met after one of the two

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

6.1.4.5 Clock Verification Timers

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

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

If the verification timer function is enabled and a failure is detected, the main clock tree is immediately switched to a working clock and an interrupt is generated to the controller. Software can then determine the course of action to take. The actual failure indication and clock switching does not clear without a write to the **CLKVCLR** register, an external reset, or a POR reset. The clock verification timers are controlled by the `PLLVER`, `BOSCVER`, and `MOSCVER` bits in the **RCC** register (see page 70).

6.1.5 System Control

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

In Run mode, the controller is actively executing code. In Sleep mode, the clocking of the device is unchanged but the controller no longer executes code (and is no longer clocked). In Deep-Sleep mode, the clocking of the device may change (depending on the Run mode clock configuration) and the controller no longer executes code (and is no longer clocked). An interrupt returns the device to Run mode from one of the sleep modes; the sleep modes are entered on request from the code.

6.2 Register Map

Table 6-1 lists the System Control registers, grouped by function. All addresses given are relative to the System Control base address of 0x400FE000.

Table 6-1. System Control Register Map (Sheet 1 of 2)

Offset	Name	Reset	Type	Description	See page
Device Identification and Capabilities					
0x000	DID0	-	RO	Device identification 0	52
0x004	DID1	-	RO	Device identification 1	53
0x008	DC0	0x70003	RO	Device capabilities 0	55
0x010	DC1	0x901F	RO	Device capabilities 1	56
0x014	DC2	0x3030011	RO	Device capabilities 2	57

Table 6-1. System Control Register Map (Sheet 2 of 2)

Offset	Name	Reset	Type	Description	See page
0x018	DC3	0x810003C0	RO	Device Capabilities 3	58
0x01C	DC4	0x7	RO	Device Capabilities 4	59
Local Control					
0x030	PBORCTL	0x00007FFD	R/W	Power-On and Brown-Out Reset Control	60
0x034	LDOPCTL	0x00000000	R/W	LDO Power Control	61
0x040	SRCR0	0x00000000	R/W	Software Reset Control 0	62
0x044	SRCR1	0x00000000	R/W	Software Reset Control 1	63
0x048	SRCR2	0x00000000	R/W	Software Reset Control 2	64
0x050	RIS	0x00000000	RO	Raw Interrupt Status	65
0x054	IMC	0x00000000	R/W	Interrupt Mask Control	66
0x058	MISC	0x00000000	R/W1C	Masked Interrupt Status and Clear	68
0x05C	RESC	-	R/W	Reset Cause	69
0x060	RCC	0x7803AC0	R/W	Run-Mode Clock Configuration	70
0x064	PLLCFG	-	RO	XTAL to PLL translation	74
System Control					
0x100	RCGC0	0x00000001	R/W	Run-Mode Clock Gating Control 0	75
0x104	RCGC1	0x00000000	R/W	Run-Mode Clock Gating Control 1	76
0x108	RCGC2	0x00000000	R/W	Run-Mode Clock Gating Control 2	77
0x110	SCGC0	0x00000001	R/W	Sleep-Mode Clock Gating Control 0	75
0x114	SCGC1	0x00000000	R/W	Sleep-Mode Clock Gating Control 1	76
0x118	SCGC2	0x00000000	R/W	Sleep-Mode Clock Gating Control 2	77
0x120	DCGC0	0x00000001	R/W	Deep-Sleep-Mode Clock Gating Control 0	75
0x124	DCGC1	0x00000000	R/W	Deep-Sleep-Mode Clock Gating Control 1	76
0x128	DCGC2	0x00000000	R/W	Deep-Sleep-Mode Clock Gating Control 2	77
0x150	CLKVCLR	0x00000000	R/W	Clock verification clear	78
0x160	LDOARST	0x00000000	R/W	Allow unregulated LDO to reset the part	79

6.3 Register Descriptions

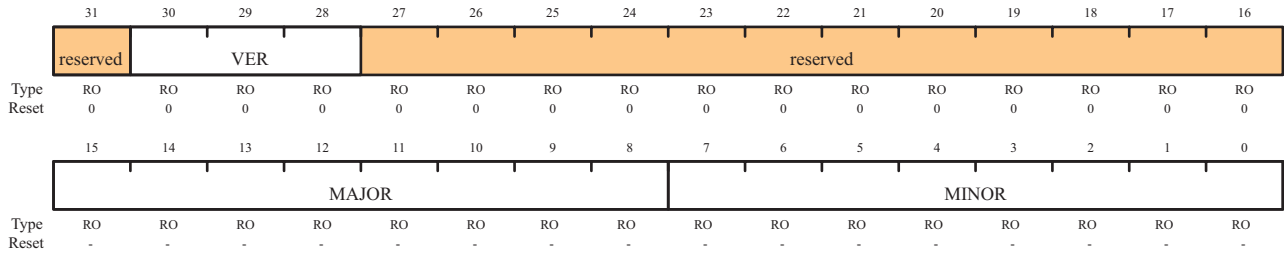
The remainder of this section lists and describes the System Control registers, in numerical order by address offset.

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

This register identifies the version of the device.

Device Identification 0 (DID0)

Offset 0x000

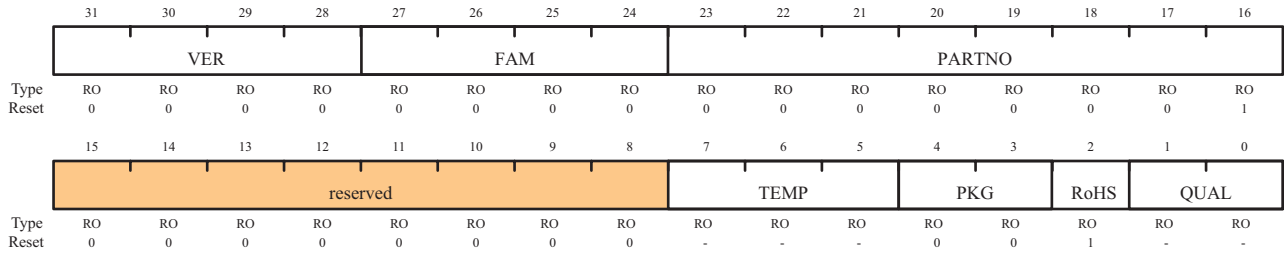


Bit/Field	Name	Type	Reset	Description
31	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
30:28	VER	RO	0	This field defines the version of the **DID0** register format: 0=Register version for the Stellaris microcontrollers
27:16	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
15:8	MAJOR	RO	-	This field specifies the major revision number of the device. 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: 0: Revision A (initial device) 1: Revision B (first revision) and so on.
7:0	MINOR	RO	-	This field specifies the minor revision number of the device. This field is numeric and is encoded as follows: 0: No changes. Major revision was most recent update. 1: One interconnect change made since last major revision update. 2: Two interconnect changes made since last major revision update. and so on.

Register 2: Device Identification 1 (DID1), offset 0x004

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

Device Identification 1 (DID1)
Offset 0x004



Bit/Field	Name	Type	Reset	Description
31:28	VER	RO	0x00	This field defines the version of the **DID1** register format: 0=Register version for the Stellaris microcontrollers
27:24	FAM	RO	0x00	Family This field provides the family identification of the device within the Luminary Micro product portfolio. The 0x00 value indicates the Stellaris family of microcontrollers.
23:16	PARTNO	RO	0x01	Part Number This field provides the part number of the device within the family. The 0x01 value indicates the LM3S101 microcontroller.
15:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:5	TEMP	RO	see table	Temperature Range This field specifies the temperature rating of the device. This field is encoded as follows: TEMP — Description 000 — Commercial temperature range (0°C to 70°C) 001 — Industrial temperature range (-40°C to 85°C) 010-111 — Reserved
4:3	PKG	RO	0x0	This field specifies the package type, where 0 indicates a 28-pin SOIC package.
2	RoHS	RO	1	RoHS-Compliance The 1 specifies the device is RoHS-compliant.

Bit/Field	Name	Type	Reset	Description
1:0	QUAL	RO	see table	This field specifies the qualification status of the device. This field is encoded as follows:

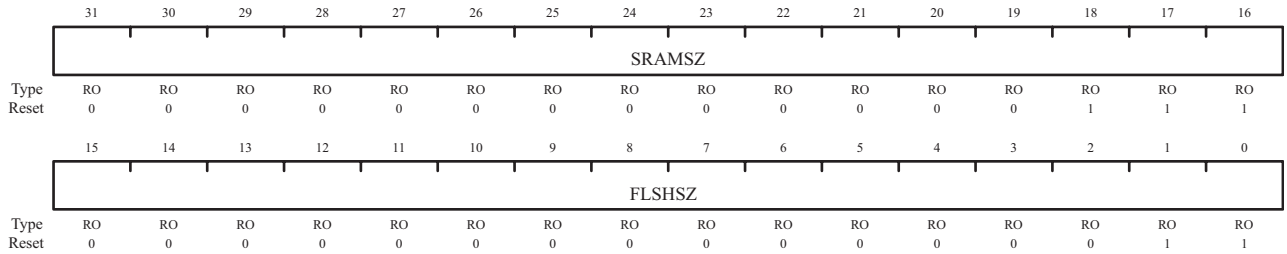
QUAL	Description
00	Engineering Sample (unqualified)
01	Pilot Production (unqualified)
10	Fully Qualified
11	Reserved

Register 3: Device Capabilities 0 (DC0), offset 0x008

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

Device Capabilities Register 0 (DC0)

Offset 0x008

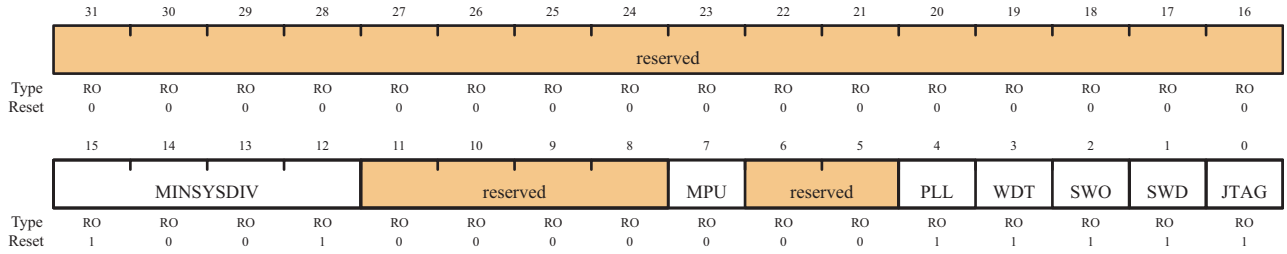


Bit/Field	Name	Type	Reset	Description
31:16	SRAMSZ	RO	0x07	Indicates the size of the on-chip SRAM. The value of 0x07 indicates 2 KB of SRAM.
15:0	FLSHSZ	RO	0x03	Indicates the size of the on-chip flash memory. The value of 0x03 indicates 8 KB of Flash.

Register 4: Device Capabilities 1 (DC1), offset 0x010

This register is predefined by the part and can be used to verify features. It also acts as a mask for write operations to the **Run-Mode Clock Gating Control 0 (RCGC0)** register (see page 75), **Sleep-Mode Clock Gating Control 0 (SCGC0)** register (see page 75), and **Deep-Sleep-Mode Clock Gating Control 0 (DCGC0)** register (see page 75).

Device Capabilities 1 (DC1)
Offset 0x010



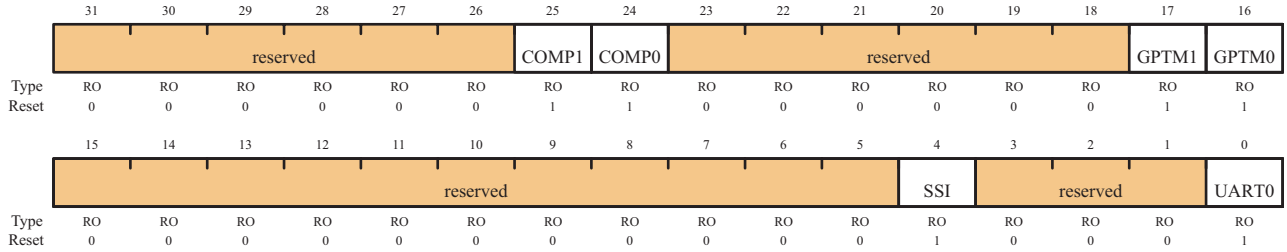
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
15:12	MINSYSDIV	RO	0x09	The reset value is hardware-dependent. The value of 0x09 specifies a 20-MHz CPU clock with a PLL divider of 10.
11:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7	MPU	RO	0x0	This bit indicates whether the Memory Protection Unit (MPU) in the Cortex-M3 is available. A 0 indicates the MPU is not available; a 1 indicates the MPU is available. See the *ARM® Cortex™-M3 Technical Reference Manual* for details on the MPU.
6:5	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
4	PLL	RO	1	A 1 in this field indicates the presence of an implemented PLL in the device.
3	WDT[a]	RO	1	A 1 in this field indicates a watchdog timer on the device.
2	SWO[a]	RO	1	A 1 in this field indicates the presence of the ARM Serial Wire Output (SWO) trace port capabilities.
1	SWD[a]	RO	1	A 1 in this field indicates the presence of the ARM Serial Wire Debug (SWD) capabilities.
0	JTAG[a]	RO	1	A 1 in this field indicates the presence of a JTAG port.

a. These bits mask the **Run-Mode Clock Gating Control 0 (RCGC0)** register (see page 113), **Sleep-Mode Clock Gating Control 0 (SCGC0)** register (see page 113), and **Deep-Sleep-Mode Clock Gating Control 0 (DCGC0)** register (see page 113). Bits that are not noted are passed as 0. ADCSP is clipped to the maximum value specified in DC1.

Register 5: Device Capabilities 2 (DC2), offset 0x014

This register is predefined by the part and can be used to verify features. It also acts as a mask for write operations to the **Run-Mode Clock Gating Control 1 (RCGC1)** register (see page 76), **Sleep-Mode Clock Gating Control 1 (SCGC1)** register (see page 76), and **Deep-Sleep-Mode Clock Gating Control 1 (DCGC1)** register (see page 76).

Device Capabilities 2 (DC2)
Offset 0x014

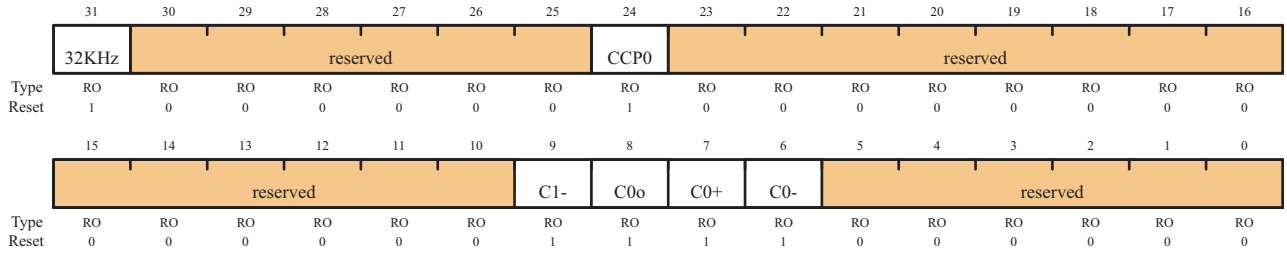


Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
25	COMP1	RO	1	A 1 in this field indicates the presence of analog comparator 1.
24	COMP0	RO	1	A 1 in this field indicates the presence of analog comparator 0.
23:18	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
17	GPTM1	RO	1	A 1 in this field indicates the presence of General-Purpose Timer module 1.
16	GPTM0	RO	1	A 1 in this field indicates the presence of General-Purpose Timer module 0.
15:5	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
4	SSI	RO	1	A 1 in this field indicates the presence of the SSI module.
3:1	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
0	UART0	RO	1	A 1 in this field indicates the presence of the UART0 module.

Register 6: Device Capabilities 3 (DC3), offset 0x018

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

Device Capabilities 3 (DC3)
Offset 0x018



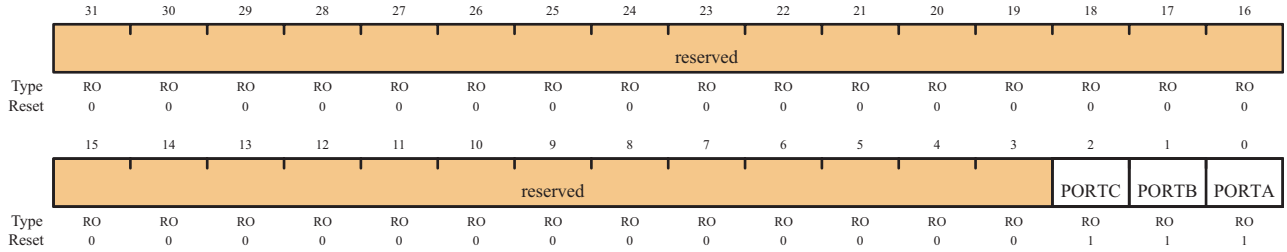
Bit/Field	Name	Type	Reset	Description
31	32KHz	RO	1	A 1 in this field indicates the presence of a 32-KHz input pin.
30:25	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
24	CCP0	RO	1	A 1 in this field indicates the presence of the Capture/ Compare/PWM pin 0.
23:10	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
9	C1-	RO	1	A 1 in this field indicates the presence of the C1- pin.
8	C0o	RO	1	A 1 in this field indicates the presence of the C0o pin.
7	C0+	RO	1	A 1 in this field indicates the presence of the C0+ pin.
6	C0-	RO	1	A 1 in this field indicates the presence of the C0- pin.
5:0	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.

Register 7: Device Capabilities 4 (DC4), offset 0x01C

This register is predefined by the part and can be used to verify features. It also acts as a mask for write operations to the **Run-Mode Clock Gating Control 2 (RCGC2)** register (see page 77), **Sleep-Mode Clock Gating Control 2 (SCGC2)** register (see page 77), and **Deep-Sleep-Mode Clock Gating Control 2 (DCGC2)** register (see page 77).

Device Capabilities 4 (DC4)

Offset 0x01C

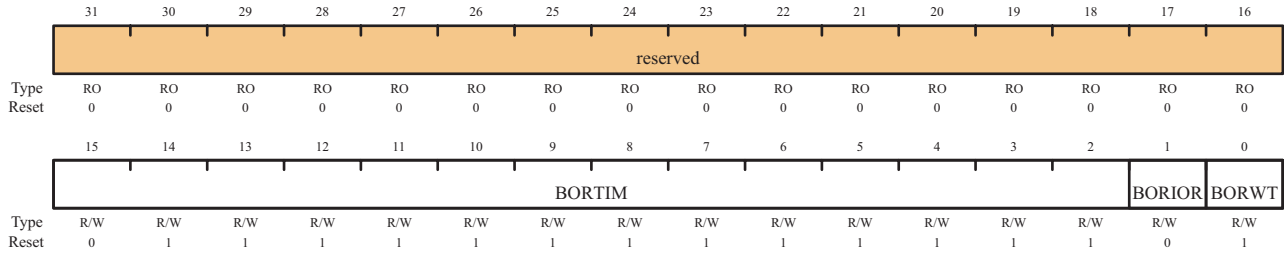


Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
2	PORTC	RO	1	A 1 in this field indicates the presence of GPIO Port C.
1	PORTB	RO	1	A 1 in this field indicates the presence of GPIO Port B.
0	PORTA	RO	1	A 1 in this field indicates the presence of GPIO Port A.

Register 8: Power-On and Brown-Out Reset Control (PBORCTL), offset 0x030

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

Power-On and Brown-Out Reset Control (PBORCTL)
Offset 0x030

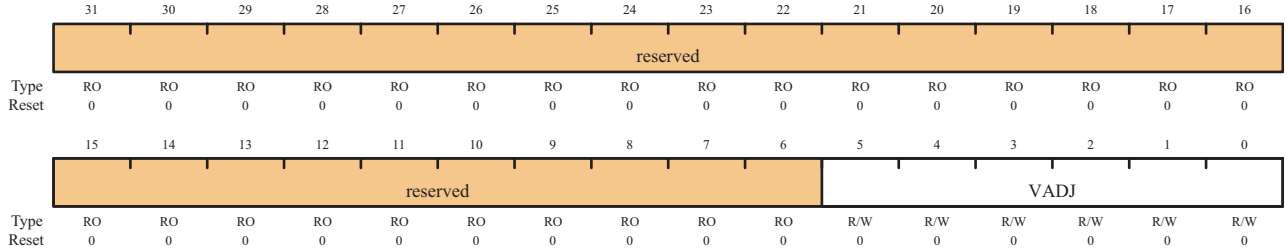


Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
15:2	BORTIM	R/W	0x1FFF	This field specifies the number of boot oscillator clocks delayed before the BOR output is resampled if the BORWT bit is set. The width of this field is derived by the t_{BOR} width of 500 µs and the boot oscillator (BOSC) frequency of 15 MHz ± 30%. At +30%, the counter value has to exceed 10,000.
1	BORIOR	R/W	0	BOR Interrupt or Reset This bit controls how a BOR event is signaled to the controller. If set, a reset is signaled. Otherwise, an interrupt is signaled.
0	BORWT	R/W	1	BOR Wait and Check for Noise This field specifies the response to a brown-out signal assertion. If BORWT is set to 1, the controller waits BORTIM BOSC periods before resampling the BOR output, and if asserted, it signals a BOR condition interrupt or reset. If the BOR resample is deasserted, the cause of the initial assertion was likely noise and the interrupt or reset is suppressed. If BORWT is 0, BOR assertions do not resample the output and any condition is reported immediately if enabled.

Register 9: LDO Power Control (LDOPCTL), offset 0x034

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

LDO Power Control (LDOPCTL)
Offset 0x034



Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
5:0	VADJ	R/W	0x0	This field specifies the value applied to the SEL_VOUT[5:0] LDO input. The programming values for the VADJ field are provided in Table 6-2.

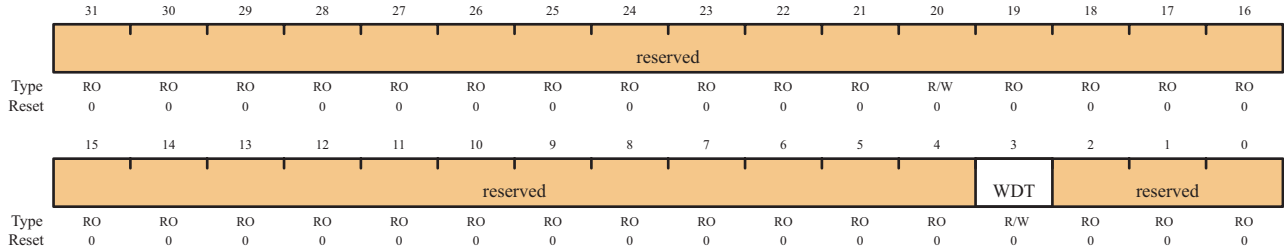
Table 6-2. VADJ to V_{OUT}

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

Register 10: Software Reset Control 0 (SRCR0), offset 0x040

Writes to this register are masked by the bits in the **Device Capabilities 1 (DC1)** register (see page 56).

Software Reset Control 0 (SRCR0)
Offset 0x040

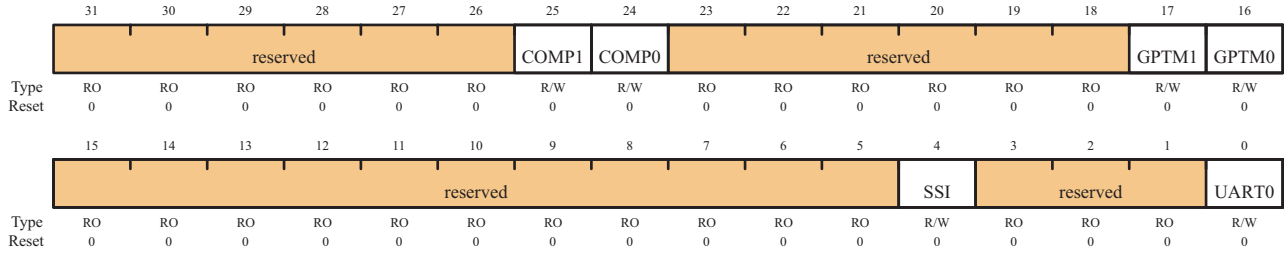


Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
3	WDT	R/W	0	Reset control for the Watchdog unit.
2:0	reserved	RO	0	Read as 0.

Register 11: Software Reset Control 1 (SRCR1), offset 0x044

Writes to this register are masked by the bits in the **Device Capabilities 2 (DC2)** register (see page 57).

Software Reset Control 1 (DC1)
Offset 0x044

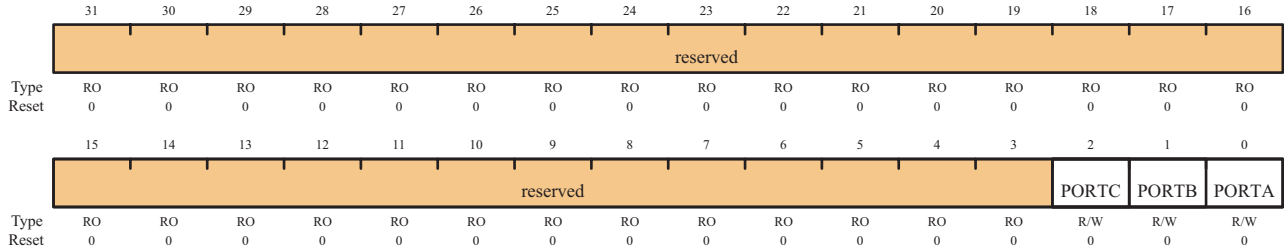


Bit/Field	Name	Type	Reset	Description
31:26	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
25	COMP1	R/W	0	Reset control for analog comparator 1.
24	COMP0	R/W	0	Reset control for analog comparator 0.
23:18	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
17	GPTM1	R/W	0	Reset control for General-Purpose Timer module 1.
16	GPTM0	R/W	0	Reset control for General-Purpose Timer module 0.
15:5	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
4	SSI	R/W	0	Reset control for the SSI units.
3:1	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
0	UART0	R/W	0	Reset control for the UART0 module.

Register 12: Software Reset Control 2 (SRCR2), offset 0x048

Writes to this register are masked by the bits in the **Device Capabilities 4 (DC4)** register (see page 59).

Software Reset Control (SRCR2)
Offset 0x048

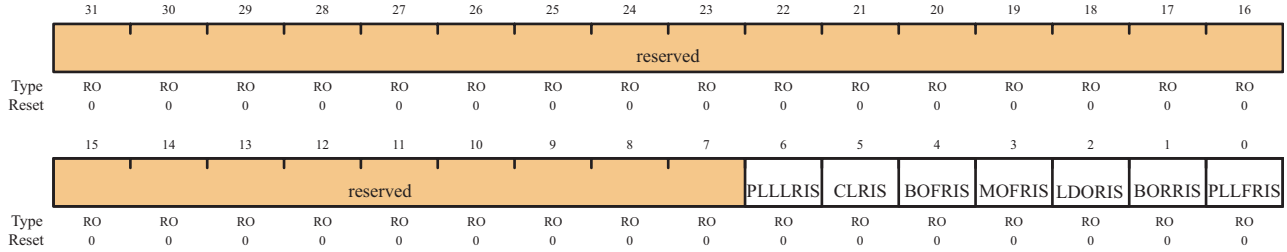


Bit/Field	Name	Type	Reset	Description
31:3	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
2	PORTC	R/W	0	Reset control for GPIO Port C.
1	PORTB	R/W	0	Reset control for GPIO Port B.
0	PORTA	R/W	0	Reset control for GPIO Port A.

Register 13: Raw Interrupt Status (RIS), offset 0x050

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

Raw Interrupt Status (RIS)
Offset 0x050

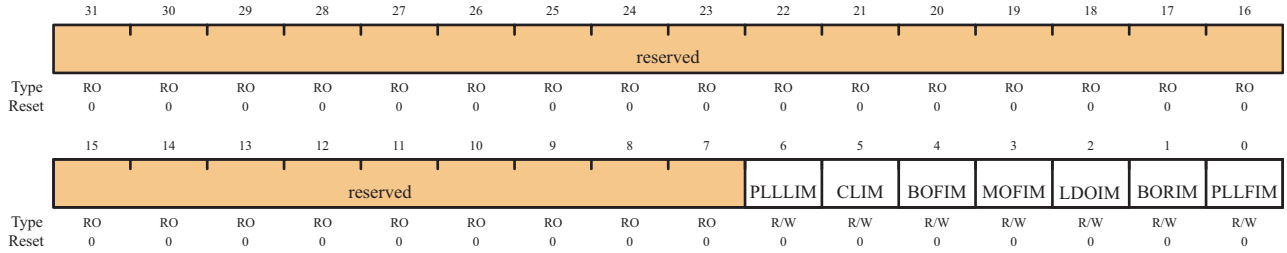


Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
6	PLLLRIS	RO	0	PLL Lock Raw Interrupt Status This bit is set when the PLL T_{READY} Timer asserts.
5	CLRIS	RO	0	Current Limit Raw Interrupt Status This bit is set if the LDO's CLE output asserts.
4	BOFRIS	RO	0	Boot Oscillator Fault Raw Interrupt Status This bit is set if a boot oscillator fault is detected.
3	MOFRIS	RO	0	Main Oscillator Fault Raw Interrupt Status This bit is set if a main oscillator fault is detected.
2	LDORIS	RO	0	LDO Power Unregulated Raw Interrupt Status This bit is set if a LDO voltage is unregulated.
1	BORRIS	RO	0	Brown-Out Reset Raw Interrupt Status This bit is the raw interrupt status for any brown-out conditions. If set, a brown-out condition was detected. An interrupt is reported if the `BORIM` bit in the **IMC** register is set and the `BORIOR` bit in the **PBORCTL** register is cleared.
0	PLLFRIS	RO	0	PLL Fault Raw Interrupt Status This bit is set if a PLL fault is detected (stops oscillating).

Register 14: Interrupt Mask Control (IMC), offset 0x054

Central location for system control interrupt masks.

Interrupt Mask Control (IMC)
Offset 0x054



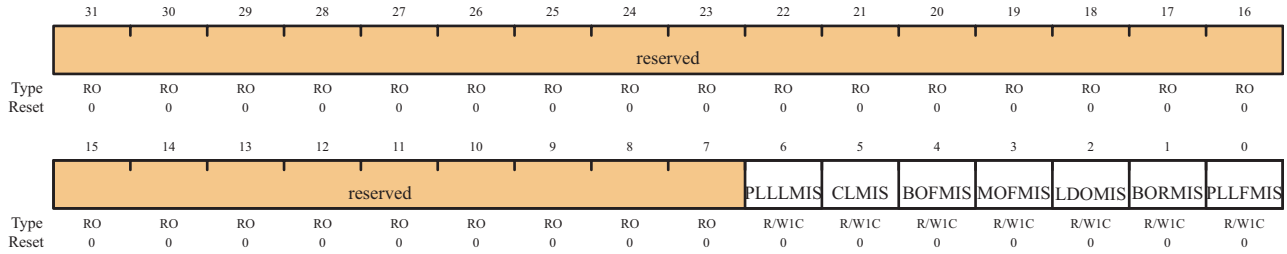
Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
6	PLLLIM	R/W	0	PLL Lock Interrupt Mask This bit specifies whether a current limit detection is promoted to a controller interrupt. If set, an interrupt is generated if PLLLRIS is set; otherwise, an interrupt is not generated.
5	CLIM	R/W	0	Current Limit Interrupt Mask This bit specifies whether a current limit detection is promoted to a controller interrupt. If set, an interrupt is generated if CLRIS is set; otherwise, an interrupt is not generated.
4	BOFIM	R/W	0	Boot Oscillator Fault Interrupt Mask This bit specifies whether a boot oscillator fault detection is promoted to a controller interrupt. If set, an interrupt is generated if BOFRIS is set; otherwise, an interrupt is not generated.
3	MOFIM	R/W	0	Main Oscillator Fault Interrupt Mask This bit specifies whether a main oscillator fault detection is promoted to a controller interrupt. If set, an interrupt is generated if MOFRIS is set; otherwise, an interrupt is not generated.
2	LDOIM	R/W	0	LDO Power Unregulated Interrupt Mask This bit specifies whether an LDO unregulated power situation is promoted to a controller interrupt. If set, an interrupt is generated if LDORIS is set; otherwise, an interrupt is not generated.

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

Register 15: Masked Interrupt Status and Clear (MISC), offset 0x058

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

Masked Interrupt Status and Clear (MISC)
Offset 0x058

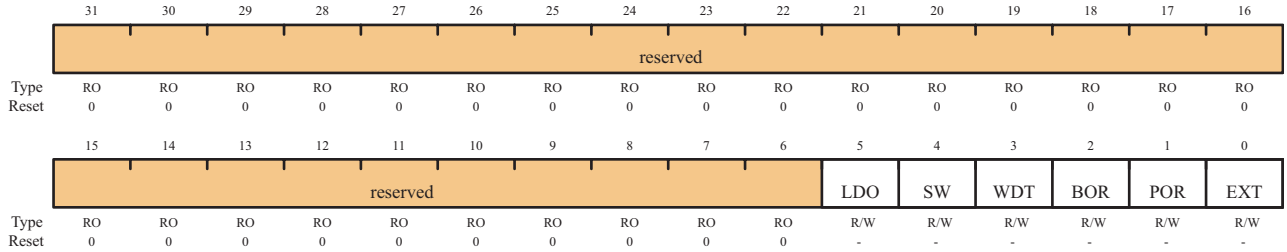


Bit/Field	Name	Type	Reset	Description
31:7	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
6	PLLLMIS	R/W1C	0	PLL Lock Masked Interrupt Status This bit is set when the PLL T_{READY} timer asserts. The interrupt is cleared by writing a 1 to this bit.
5	CLMIS	R/W1C	0	Current Limit Masked Interrupt Status This bit is set if the LDO's CLE output asserts. The interrupt is cleared by writing a 1 to this bit.
4	BOFMIS	R/W1C	0	Boot Oscillator Fault Masked Interrupt Status This bit is set if a boot oscillator fault is detected. The interrupt is cleared by writing a 1 to this bit.
3	MOFMIS	R/W1C	0	Main Oscillator Fault Masked Interrupt Status This bit is set if a main oscillator fault is detected. The interrupt is cleared by writing a 1 to this bit.
2	LDOMIS	R/W1C	0	LDO Power Unregulated Masked Interrupt Status This bit is set if LDO power is unregulated. The interrupt is cleared by writing a 1 to this bit.
1	BORMIS	R/W1C	0	Brown-Out Reset Masked Interrupt Status This bit is the masked interrupt status for any brown-out conditions. If set, a brown-out condition was detected. An interrupt is reported if the `BORIM` bit in the **IMC** register is set and the `BORIOR` bit in the **PBORCTL** register is cleared. The interrupt is cleared by writing a 1 to this bit.
0	PLLFMIS	R/W1C	0	PLL Fault Masked Interrupt Status This bit is set if a PLL fault is detected (stops oscillating). The interrupt is cleared by writing a 1 to this bit.

Register 16: Reset Cause (RESC), offset 0x05C

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

Reset Cause (RESC)
Offset 0x05C

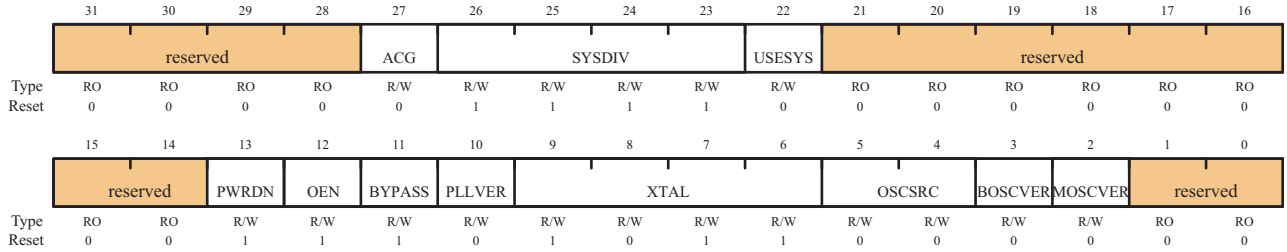


Bit/Field	Name	Type	Reset	Description
31:6	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
5	LDO	R/W	-	When set to 1, LDO power OK lost is the cause of the reset event.
4	SW	R/W	-	When set to 1, a software reset is the cause of the reset event.
3	WDT	R/W	-	When set to 1, a watchdog reset is the cause of the reset event.
2	BOR	R/W	-	When set to 1, a brown-out reset is the cause of the reset event.
1	POR	R/W	-	When set to 1, a power-on reset is the cause of the reset event.
0	EXT	R/W	-	When set to 1, an external reset ($\overline{\text{RST}}$ assertion) is the cause of the reset event.

Register 17: Run-Mode Clock Configuration (RCC), offset 0x060

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

Run-Mode Clock Configuration (RCC)
Offset 0x060



Bit/Field	Name	Type	Reset	Description
31:28	Reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
27	ACG	R/W	0	Auto Clock Gating This bit specifies whether the system uses the **Sleep-Mode Clock Gating Control (SCGCn)** registers (see page 75) and **Deep-Sleep-Mode Clock Gating Control (DCGCn)** registers (see page 75) 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 (see page 75) 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 (200 MHz). When reading the **Run-Mode Clock Configuration (RCC)** register (see page 70), the `SYSDIV` value will be `MINSYSDIV` if a lower divider was requested and the PLL is being used. This lower value will be allowed to divide a non-PLL source.
22	USESYS	R/W	0	Use the system clock divider as the source for the system clock. The system clock divider is forced to be used when the PLL is selected as the source.
21:14	reserved	RO	1	Read as 1.
13	PWRDN	R/W	1	PLL Power Down This bit connects to the PLL PWRDN input. The reset value of 1 powers down the PLL. See Table 6-4 on page 73 for PLL mode control.
12	OEN	R/W	1	PLL Output Enable This bit specifies whether the PLL output driver is enabled. If cleared, the driver transmits the PLL clock to the output. Otherwise, the PLL clock does not oscillate outside the PLL module. **Note:** Both PWRDN and OEN must be cleared to run the PLL.
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.

Binary Value	Divisor (BYPASS=1)	Frequency (BYPASS=0)
0000-1000	reserved	reserved
1001	/10	20 MHz
1010	/11	18.18 MHz
1011	/12	16.67 MHz
1100	/13	15.38 MHz
1101	/14	14.29 MHz
1110	/15	13.33 MHz
1111	/16	12.5 MHz (default)

Bit/Field	Name	Type	Reset	Description
10	PLLVER	R/W	0x0	PLL Verification This bit controls the PLL verification timer function. If set, the verification timer is enabled and an interrupt is generated if the PLL becomes inoperative. Otherwise, the verification timer is not enabled.
9:6	XTAL	R/W	0xB	This field specifies the crystal value attached to the main oscillator. The encoding for this field is provided in Table 6-3 on page 72.
Oscillator-Related Bits				
5:4	OSCSRC	R/W	0x0	Picks among the four input sources for the OSC leg. The values are: Value — Input Source 00 — Main oscillator 01 — Boot oscillator 10 — Boot oscillator / 4 (this is necessary if used as input to PLL) 11 — reserved
3	BOSCVER	R/W	0x0	This bit controls the boot oscillator verification timer function. If set, the verification timer is enabled and an interrupt is generated if the timer becomes inoperative. Otherwise, the verification timer is not enabled.
2	MOSCVER	R/W	0x0	This bit controls the main oscillator verification timer function. If set, the verification timer is enabled and an interrupt is generated if the timer becomes inoperative. Otherwise, the verification timer is not enabled.
1:0	reserved	RO	0x0	Reserved bits return an indeterminate value, and should never be changed.

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

Crystal Number (XTAL Binary Value)	Crystal Frequency (MHz)
0000-0011	reserved
0100	3.579545 MHz
0101	3.6864 MHz
0110	4 MHz
0111	4.096 MHz
1000	4.9152 MHz
1001	5 MHz

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

Crystal Number (XTAL Binary Value)	Crystal Frequency (MHz)
1010	5.12 MHz
1011	6 MHz (reset value)
1100	6.144 MHz
1101	7.3728 MHz
1110	8 MHz
1111	8.192 MHz

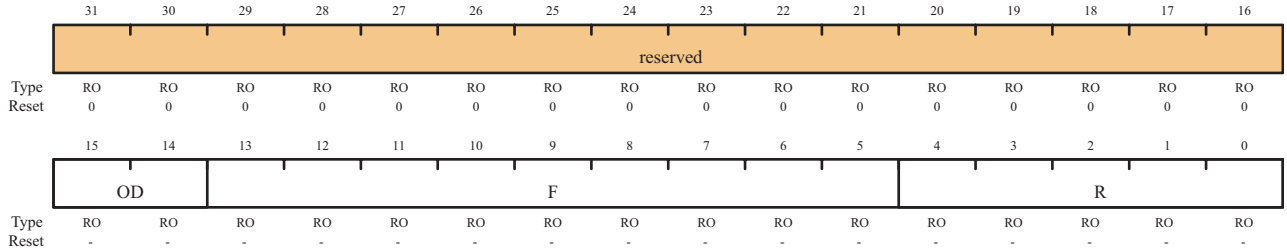
Table 6-4. PLL Mode Control

PWRDN	OEN	Mode
1	X	Power down
0	0	Normal

Register 18: 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 70).

XTAL to PLL Translation (PLLCFG)
Offset 0x064



Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
15:14	OD	RO	-	This field specifies the value supplied to the PLL's OD input.
13:5	F	RO	-	This field specifies the value supplied to the PLL's F input.
4:0	R	RO	-	This field specifies the value supplied to the PLL's R input.

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

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

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

These registers control 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). 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 (see page 70) specifies that the system uses sleep modes.

The bit diagram for these three registers is shown below. Bits have the same definitions as **DC1** (see page 56) and use **DC1** as the mask.

Run-Mode, Sleep-Mode and Deep-Sleep-Mode Clock Gating Control 0 (RCGC0, SCG0, and DCGC0)
Offset 0x100, 0x110, 0x120

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

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

Register 22: Run-Mode Clock Gating Control 1 (RCGC1), offset 0x104

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

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

These registers control 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). 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 (see page 70) specifies that the system uses sleep modes.

The bit diagram for these three registers is shown below. Bits have the same definitions as **DC2** (see page 57) and use **DC2** as the mask.

Run-Mode, Sleep-Mode, and Deep-Sleep-Mode Clock Gating Control 1 (RCGC1, SCGC1, and DCGC1)
Offset 0x104, 0x114, and 0x124

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved						COMP1	COMP0	reserved						GPTM1	GPTM0
Type	RO	RO	RO	RO	RO	RO	R/W	R/W	RO	RO	RO	RO	RO	RO	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											SSI	reserved			UART0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	RO	RO	RO	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

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

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

These registers control 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). 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 (see page 70) specifies that the system uses sleep modes.

The bit diagram for these three registers is shown below. Bits have the same definitions as **DC4** (see page 59) and use **DC4** as the mask.

Run-Mode, Sleep-Mode, and Deep-Sleep-Mode Clock Gating Control 2 (RCGC2, SCGC2, and DCGC2)
Offset 0x108, 0x118, and 0x128

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

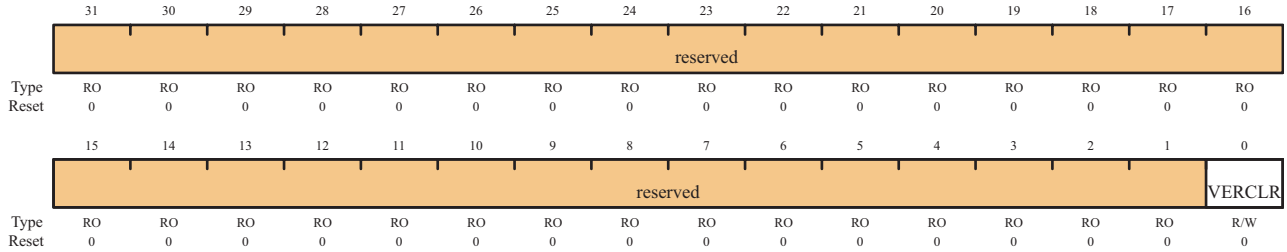
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved													PORTC	PORTB	PORTA
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Register 28: Clock Verification Clear (CLKVCLR), offset 0x150

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

Clock Verification Clear (CLKVCLR)

Offset 0x150



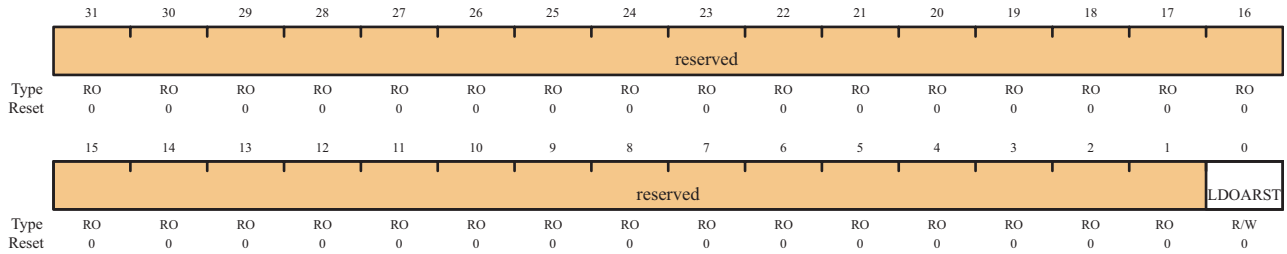
Bit/Field	Name	Type	Reset	Description
31:1	Reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
0	VERCLR	R/W	0	Clear clock verification faults.

Register 29: Allow Unregulated LDO to Reset the Part (LDOARST), offset 0x160

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

Allow Unregulated LDO to Reset the Part (LDOARST)

Offset 0x160



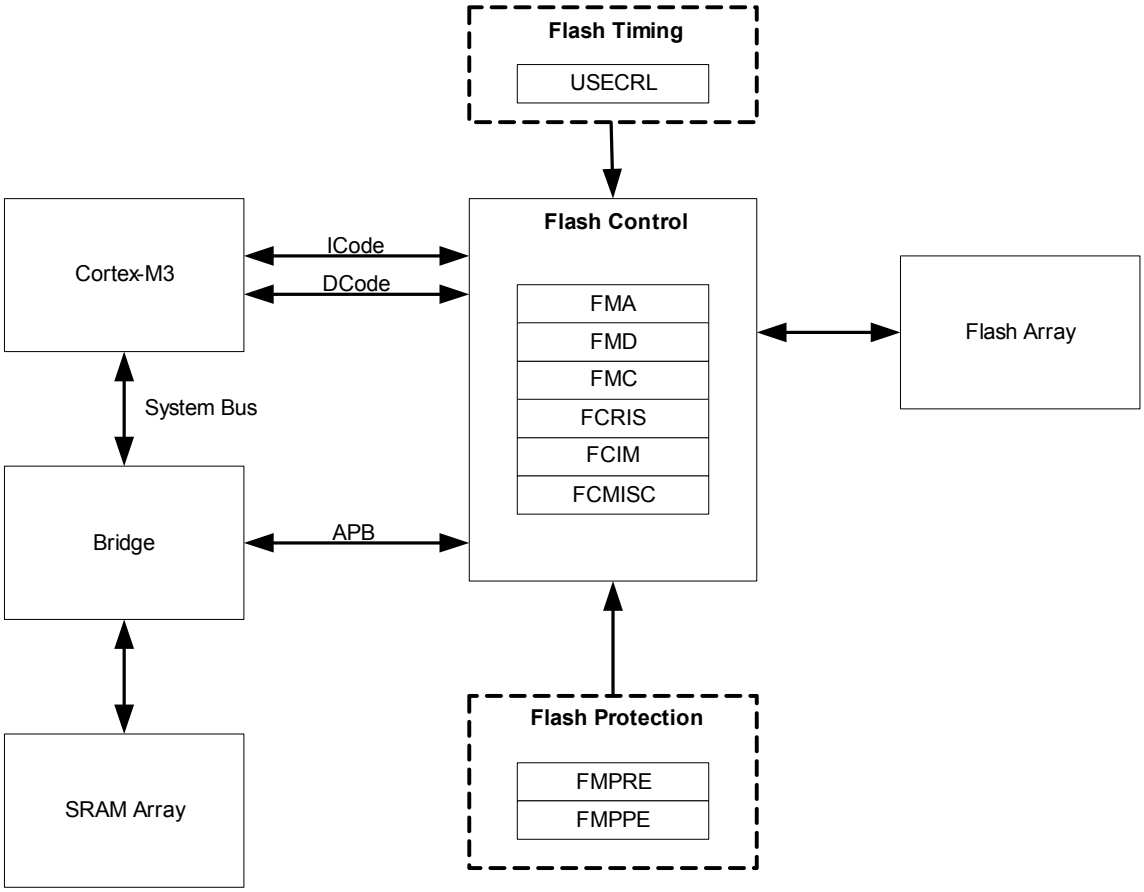
Bit/Field	Name	Type	Reset	Description
31:1	Reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
0	LDOARST	R/W	0	Set to 1 to allow unregulated LDO output to reset the part.

7 Internal Memory

The LM3S101 comes with 2 KB of bit-banded SRAM and 8 KB of flash memory. The flash controller provides a user-friendly interface, making flash programming a simple task. Flash protection can be applied to the flash memory on a 2-KB block basis.

7.1 Block Diagram

Figure 7-1. Flash Block Diagram



7.2 Functional Description

This section describes the functionality of both memories.

7.2.1 SRAM Memory

The internal SRAM of the Stellaris devices is located at address 0x20000000 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.

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 0x20001000 is to be modified, the bit-band alias is calculated as:

```
0x22000000 + (0x1000 * 32) + (3 * 4) = 0x2202000C
```

With the alias address calculated, an instruction performing a read/write to address 0x2202000C allows direct access to only bit 3 of the byte at address 0x20001000.

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

7.2.2 Flash Memory

The flash is organized as a set of 1-KB blocks that can be individually erased. Erasing a block causes the entire contents of the block to be reset to all 1s. 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.

7.2.2.1 Flash Memory Timing

The timing for the flash is automatically handled by the flash controller. However, in order to do so, it must know the clock rate of the system in order to time its internal signals properly. The number of clock cycles per microsecond must be provided to the flash controller for it to accomplish this timing. It is software's responsibility to keep the flash controller updated with this information via the **U Second Reload (USECRL)** register (see page 85).

On reset, **USECRL** is loaded with a value that will configure the flash timing so that it works with the selected crystal value. If software changes the system operating frequency, the new operating frequency 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 must be written to the **USECRL** register.

7.2.2.2 Flash Memory Protection

The user is provided two forms of flash protection per 2-KB flash blocks in two 32-bit wide registers. The protection policy for each form is controlled by individual bits (per policy per block) in the **FMPPE** and **FMPRE** registers (see page 84).

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

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

Table 7-1. Flash Protection Policy Combinations

FMPPE	FMPRE	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.

Table 7-1. Flash Protection Policy Combinations

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

7.2.2.3 Flash Memory Programming

Writing the flash memory requires that the code be executed out of SRAM to avoid corrupting or interrupting the bus timing. Flash pages can be erased on a page basis (1 KB in size), or by performing a mass erase of the entire flash.

All erase and program operations are performed using the **Flash Memory Address (FMA)**, **Flash Memory Data (FMD)** and **Flash Memory Control (FMC)** registers. See section 7.3 for examples.

7.3 Initialization and Configuration

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

7.3.1 Changing Flash Protection Bits

As discussed in Section 7.2.2.2, changes to the protection bits must be committed before they take effect. The sequence to change and commit a bit in software is as follows:

1. The **Flash Memory Protection Read Enable (FMPRE)** and **Flash Memory Protection Program Enable (FMPPE)** registers are written, changing the intended bit(s). The action of these changes can be tested by software while in this state.
2. The **Flash Memory Address (FMA)** register (see page 86) bit 0 is set to 1 if the **FMPPE** register is to be committed; otherwise, a 0 commits the **FMPRE** register.
3. The **Flash Memory Control (FMC)** register (see page 88) is written with the COMT bit set. This initiates a write sequence and commits the changes.

7.3.2 Flash Programming

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

The flash is programmed using the following sequence:

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 0xA4420001) to the **FMC** register.
4. Poll the **FMC** register until the `WRITE` bit is cleared.

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 0xA4420002) to the **FMC** register.
3. Poll the **FMC** register until the `ERASE` bit is cleared.

To perform a mass erase of the flash:

1. Write the flash write key and the `MERASE` bit (a value of 0xA4420004) to the **FMC** register.
2. Poll the **FMC** register until the `MERASE` bit is cleared.

7.4 Register Map

Table 7-2 lists the Flash memory and control registers. All addresses given are relative to the Flash control base address of 0x400FD000, except for **FMPRE** and **FMPPE**, which are relative to the System Control base address of 0x400FE000.

Table 7-2. Flash Register Map

Offset	Name	Reset	Type	Description	See page
0x130[a]	FMPRE	0x0F	R/W0	Flash memory read protect	84
0x134[a]	FMPPE	0x0F	R/W0	Flash memory program protect	84
0X140[a]	USECRL	0x13	R/W	U second reload	85
0x000	FMA	0x00000000	R/W	Flash memory address	86
0x004	FMD	0x00000000	R/W	Flash memory data	87
0x008	FMC	0x00000000	R/W	Flash memory control	88
0x00C	FCRIS	0x00000000	RO	Flash controller raw interrupt status	90
0x010	FCIM	0x00000000	R/W	Flash controller interrupt mask	91
0x014	FCMISC	0x00000000	R/W1C	Flash controller masked interrupt status and clear	92

a. Relative to System Control base address of 0x400FE000.

7.5 Register Descriptions

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

Register 1: Flash Memory Protection Read Enable (FMPRE), offset 0x130

Register 2: Flash Memory Protection Program Enable (FMPPE), offset 0x134

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

These registers store the read-only (**FMPRE**) and execute-only (**FMPPE**) protection bits for each 2 KB flash block. This register is loaded during the power-on reset sequence.

The factory settings for the **FMPRE** and **FMPPE** 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. 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 “Flash Memory Protection” on page 81.

Flash Memory Protection Read Enable and Program Enable (FMPRE and FMPPE)

Offset 0x130 and 0x134

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												Block3	Block2	Block1	Block0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W0	R/W0	R/W0	R/W0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
3:0	Block3-Block0	R/W0	0x0F	Enable 2-KB flash blocks to be written or erased (**FMPPE** register), or executed or read (**FMPRE** register). The policies may be combined as shown in Table 7-1 on page 81.

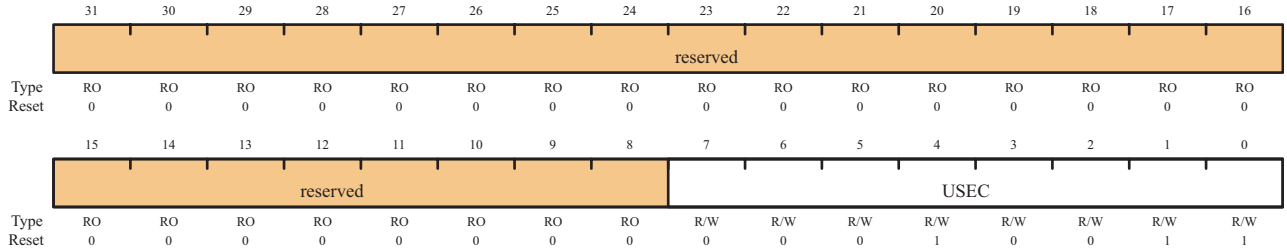
Register 3: U Second Reload (USECRL), offset 0x140

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

This register is provided as a means of creating a 1 usec tick divider reload value for the flash controller. The internal flash has specific minimum and maximum requirement 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)

Offset 0x140



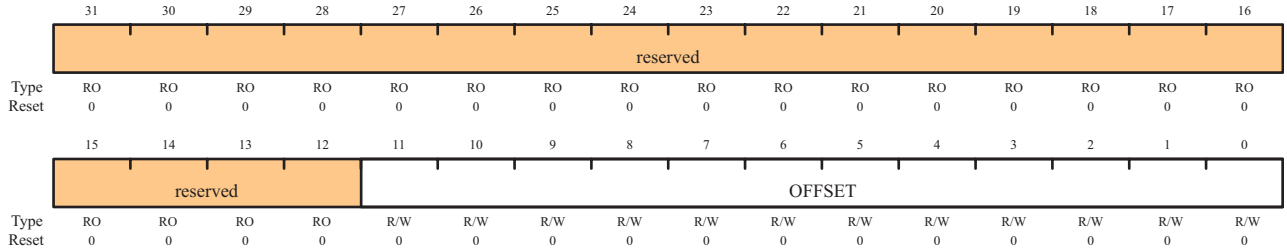
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	USEC	R/W	0x13	MHz -1 of the controller clock when the flash is being erased or programmed. USEC should be set to 0x13 (19 MHz) whenever the flash is being erased or programmed.

Register 4: 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)

Offset 0x000

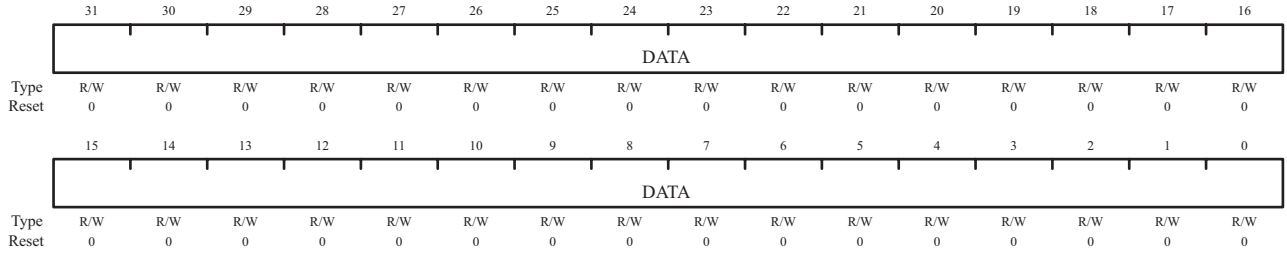


Bit/Field	Name	Type	Reset	Description
31:12	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
11:0	OFFSET	R/W	0	Address offset in flash where operation is performed.

Register 5: 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)
Offset 0x004



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

Register 6: Flash Memory Control (FMC), offset 0x008

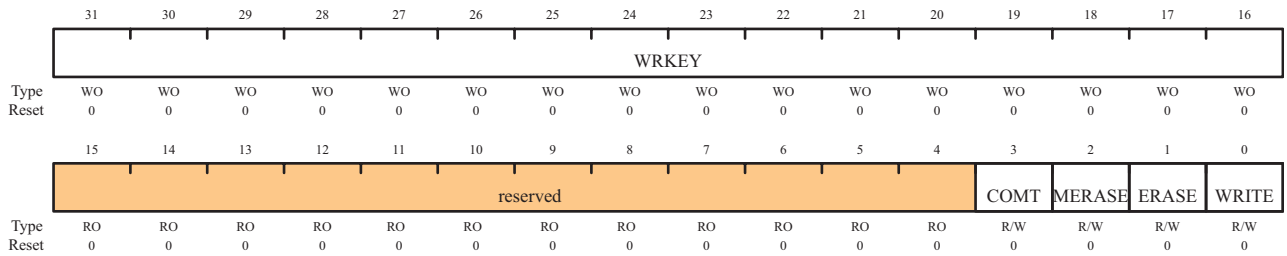
When this register is written, the flash controller initiates the appropriate access cycle for the location specified by the **Flash Memory Address (FMA)** register (see page 86). If the access is a write access, the data contained in the **Flash Memory Data (FMD)** register (see page 87) is written.

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)

Offset 0x008



Bit/Field	Name	Type	Reset	Description
31:16	WRKEY	WO	0	This field contains a write key, which is used to minimize the incidence of accidental flash writes. The value 0xA442 must be written into this field for a write to occur. Writes to the **FMC** register without this WRKEY value are ignored. A read of this field returns the value 0.
15:4	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
3	COMT	R/W	0	Commit (write) of register value to non-volatile 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.

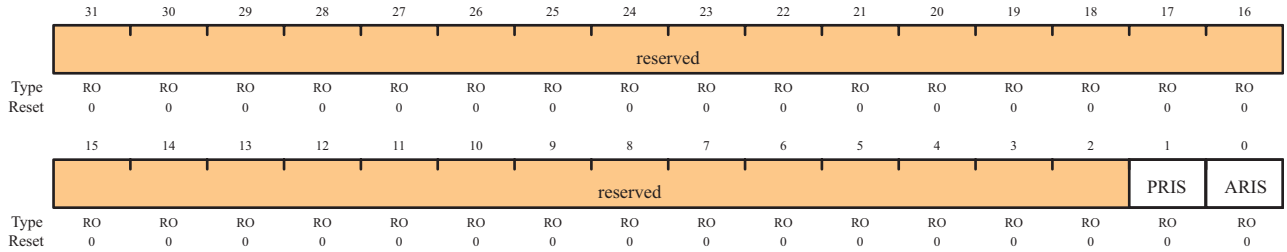
Bit/Field	Name	Type	Reset	Description
1	ERASE	R/W	0	Erase a page of flash memory If this bit is set, the page of flash main memory as specified by the contents of **FMA** is erased. A write of 0 has no effect on the state of this bit. If read, the state of the previous erase access is provided. If the 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 7: 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)

Offset 0x00C



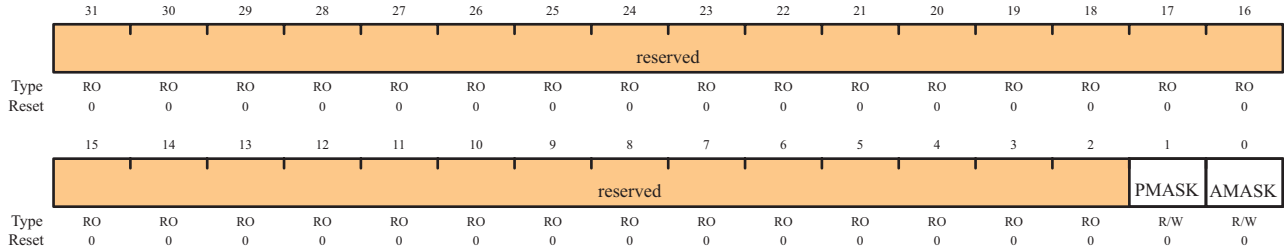
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
1	PRIS	RO	0	Programming Raw Interrupt Status This bit indicates the current state of the programming cycle. If set, the programming cycle completed; if cleared, the programming cycle has not completed. Programming cycles are either write or erase actions generated through the **Flash Memory Control (FMC)** register bits (see page 88).
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 (FMPRE)** and **Flash Memory Protection Program Enable (FMPPE)** register (see page 84). Otherwise, no access has tried to improperly access the flash.

Register 8: Flash Controller Interrupt Mask (FCIM), offset 0x010

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

Flash Controller Interrupt Mask (FCIM)

Offset 0x010



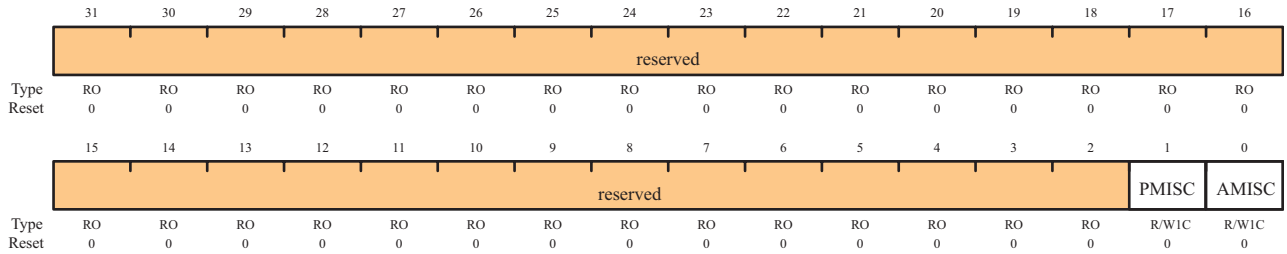
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
1	PMASK	R/W	0	Programming Interrupt Mask This bit controls the reporting of the programming raw interrupt status to the controller. If set, a programming-generated interrupt 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 9: 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)

Offset 0x014



Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
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 90) 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 General-Purpose Input/Outputs (GPIOs)

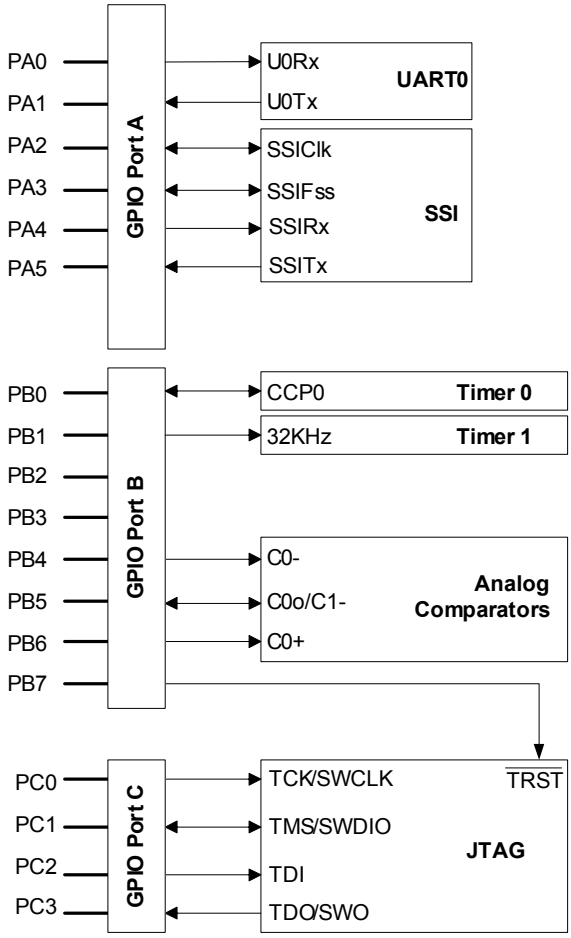
The GPIO module is composed of three physical GPIO blocks, each corresponding to an individual GPIO port (Port A, Port B, and Port C). The GPIO module is FiRM-compliant and supports 2 to 18 programmable input/output pins, depending on the peripherals being used.

The GPIO module has the following features:

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

8.1 Block Diagram

Figure 8-1. GPIO Module Block Diagram

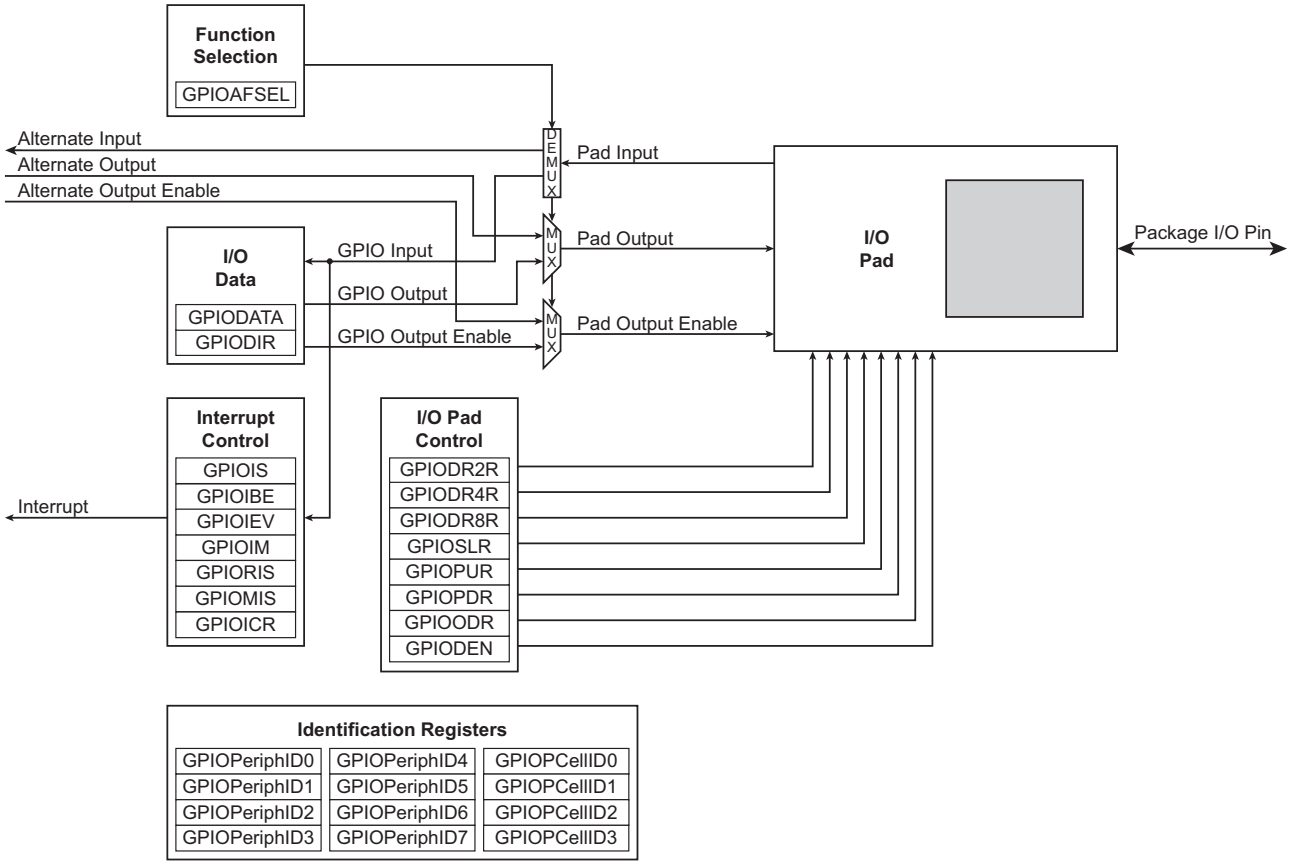


8.2 Functional Description

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

Each GPIO port is a separate hardware instantiation of the same physical block (see Figure 8-2). The LM3S101 microcontroller contains three of these physical GPIO blocks.

Figure 8-2. GPIO Port Block Diagram



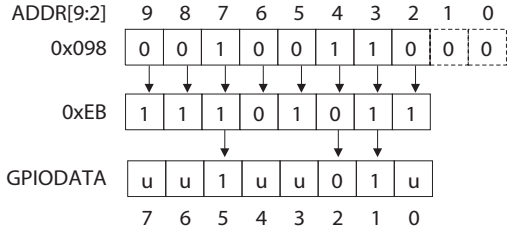
8.2.1 Data Register Operation

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

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

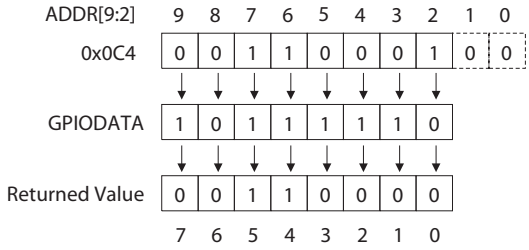
For example, writing a value of 0xEB to the address GPIODATA + 0x098 would yield as shown in Figure 8-3, where u is data unchanged by write.

Figure 8-3. 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 8-4.

Figure 8-4. GPIODATA Read Example



8.2.2 Data Direction

The **GPIO Direction (GPIODIR)** register (see page 101) is used to configure each individual pin as an input or output.

8.2.3 Interrupt Operation

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 102)
- **GPIO Interrupt Both Edges (GPIOIBE)** register (see page 103)
- **GPIO Interrupt Event (GPIOIEV)** register (see page 104)

Interrupts are enabled/disabled via the **GPIO Interrupt Mask (GPIOIM)** register (see page 105). 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 pages 106 and 107). As the name implies, the **GPIOMIS** register only shows interrupt conditions that are allowed to be passed to the controller. The **GPIORIS** register indicates that a GPIO pin meets the conditions for an interrupt, but has not necessarily been sent to the controller.

Interrupts are cleared by writing a 1 to the **GPIO Interrupt Clear (GPIOICR)** register (see page 108).

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

8.2.4 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 109), the pin state is controlled by its alternate function (that is, the peripheral). Software control corresponds to GPIO mode, where the **GPIODATA** register is used to read/write the corresponding pins.

8.2.5 Pad Configuration

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

8.2.6 Identification

The identification registers configured at reset allow software to detect and identify the module as a GPIO block. The identification registers include the **GPIOPeriphID0**-**GPIOPeriphID7** registers as well as the **GPIOPCellID0**-**GPIOPCellID3** registers.

8.3 Initialization and Configuration

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

Table 8-1. Pad Configuration Examples

Configuration	Register Bit Value[a]									
	GPIOAFSEL	GPIODIR	GPIOODR	GPIODEN	GPIOPUR	GPIOPDR	GPIODR2R	GPIODR4R	GPIODR8R	GPIOSLR
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	?	?	?	?
Analog Input (Comparator)	0	0	0	0	0	0	X	X	X	X
Digital Output (Comparator)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (UART)	1	X	0	1	?	?	?	?	?	?
Digital Input/Output (SSI)	1	X	0	1	?	?	?	?	?	?
Digital Output (Timer PWM)	1	X	0	1	?	?	?	?	?	?
Digital Input (Timer CCP)	1	X	0	1	?	?	X	X	X	X

a. X=Ignored (don't care bit)
?=Can be either 0 or 1, depending on the configuration

Table 8-2. Interrupt Configuration Example

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

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

8.4 Register Map

Table 8-2 lists the GPIO registers. All addresses given are relative to that GPIO port's base address:

- GPIO Port A: 0x40004000
- GPIO Port B: 0x40005000
- GPIO Port C: 0x40006000

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 (see Figure 8-1 on page 94). In those cases, writing to those unconnected bits has no effect and reading those unconnected bits returns no meaningful data.

Table 8-3. GPIO Register Map

Offset	Name	Reset	Type	Description	See page
0x000	GPIODATA	0x00000000	R/W	Data	100
0x400	GPIODIR	0x00000000	R/W	Data direction	101
0x404	GPIOIS	0x00000000	R/W	Interrupt sense	102
0x408	GPIOIBE	0x00000000	R/W	Interrupt both edges	103
0x40C	GPIOIEV	0x00000000	R/W	Interrupt event	104
0x410	GPIOIM	0x00000000	R/W	Interrupt mask enable	105
0x414	GPIORIS	0x00000000	RO	Raw interrupt status	106
0x418	GPIOMIS	0x00000000	RO	Masked interrupt status	107
0x41C	GPIOICR	0x00000000	W1C	Interrupt clear	108
0x420	GPIOAFSEL	see note[a]	R/W	Alternate function select	109

Table 8-3. GPIO Register Map

Offset	Name	Reset	Type	Description	See page
0x500	GPIODR2R	0x000000FF	R/W	2-mA drive select	110
0x504	GPIODR4R	0x00000000	R/W	4-mA drive select	111
0x508	GPIODR8R	0x00000000	R/W	8-mA drive select	112
0x50C	GPIOODR	0x00000000	R/W	Open drain select	113
0x510	GPIOPUR	0x000000FF	R/W	Pull-up select	114
0x514	GPIOPDR	0x00000000	R/W	Pull-down select	115
0x518	GPIOSLR	0x00000000	R/W	Slew rate control select	116
0x51C	GPIODEN	0x000000FF	R/W	Digital input enable	117
0xFD0	GPIOPeriphID4	0x00000000	RO	Peripheral identification 4	118
0xFD4	GPIOPeriphID5	0x00000000	RO	Peripheral identification 5	119
0xFD8	GPIOPeriphID6	0x00000000	RO	Peripheral identification 6	120
0xFDC	GPIOPeriphID7	0x00000000	RO	Peripheral identification 7	121
0xFE0	GPIOPeriphID0	0x00000061	RO	Peripheral identification 0	122
0xFE4	GPIOPeriphID1	0x00000000	RO	Peripheral identification 1	123
0xFE8	GPIOPeriphID2	0x00000018	RO	Peripheral identification 2	124
0xFEC	GPIOPeriphID3	0x00000001	RO	Peripheral identification 3	125
0xFF0	GPIOPCellID0	0x0000000D	RO	GPIO PrimeCell identification 0	126
0xFF4	GPIOPCellID1	0x000000F0	RO	GPIO PrimeCell identification 1	127
0xFF8	GPIOPCellID2	0x00000005	RO	GPIO PrimeCell identification 2	128
0xFFC	GPIOPCellID3	0x000000B1	RO	GPIO PrimeCell identification 3	129

a. The default reset value for the **GPIOAFSEL** register is 0x00000000 for all GPIO pins, with the exception of the five JTAG pins (PB7 and PC[3:0]. These five pins default to JTAG functionality. Because of this, the default reset value of **GPIOAFSEL** for GPIO Port B is 0x00000080 while the default reset value of **GPIOAFSEL** for Port C is 0x0000000F.

8.5 Register Descriptions

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

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

The **GPIODATA** register is the data register. In software control mode, values written in the **GPIODATA** register are transferred onto the GPIO port pins if the respective pins have been configured as outputs through the **GPIO Direction (GPIODIR)** register (see page 101).

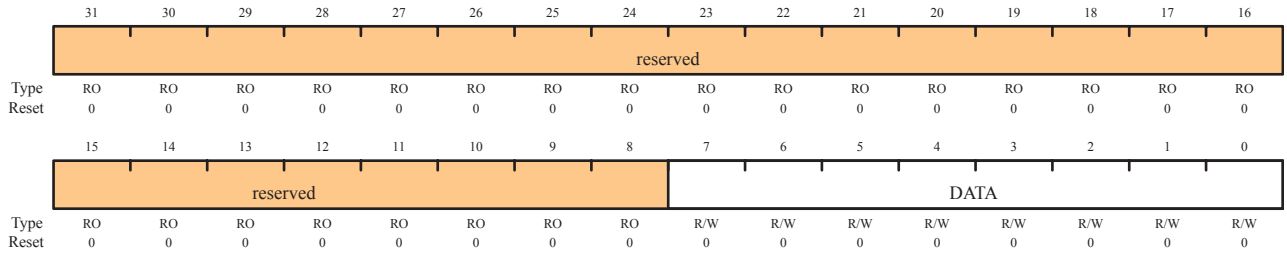
In order to write to **GPIODATA**, the corresponding bits in the mask, resulting from the address bus bits [9:2], must be High. Otherwise, the bit values remain unchanged by the write.

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 output, 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)

Offset 0x000



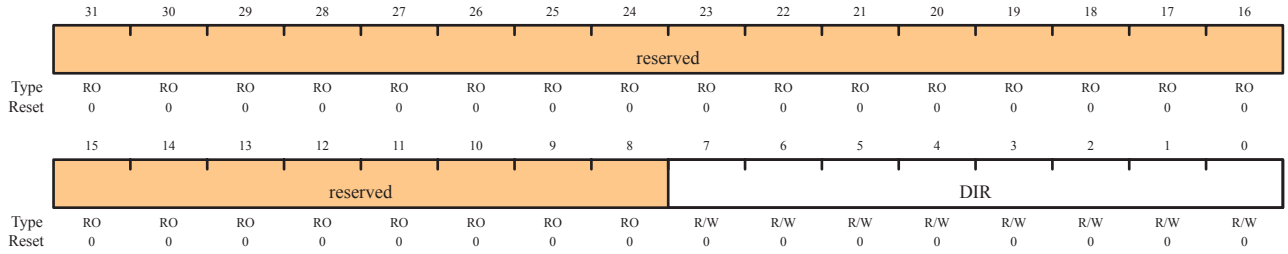
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	DATA	R/W	0	GPIO Data This register is virtually mapped to 256 locations in the address space. To facilitate the reading and writing of data to these registers by independent drivers, the data read from and the data written to the registers are masked by the eight address lines `ipaddr[9:2]`. Reads from this register return its current state. Writes to this register only affect bits that are not masked by `ipaddr[9:2]` and are configured as outputs. See “Data Register Operation” on page 95 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)

Offset 0x400



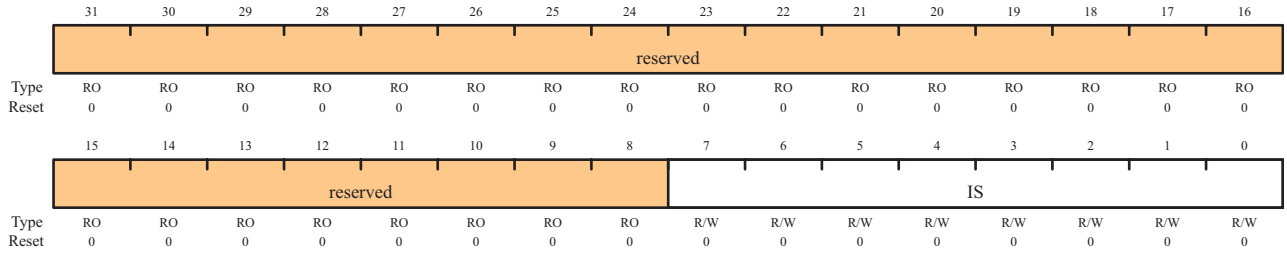
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	DIR	R/W	0x00	GPIO Data Direction 0: Pins are inputs 1: Pins are outputs

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

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

GPIO Interrupt Sense (GPIOIS)

Offset 0x404



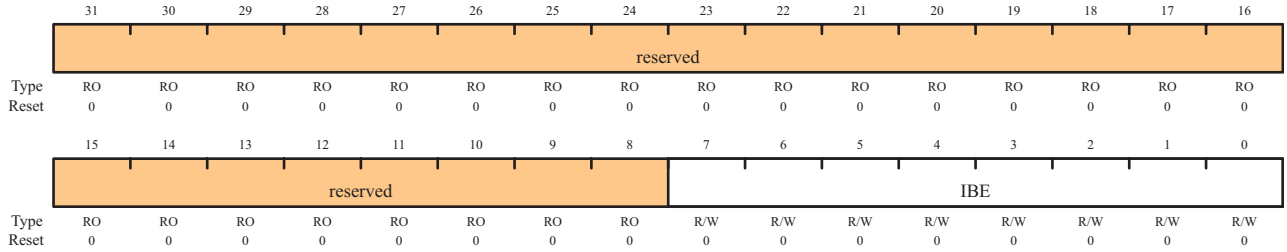
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	IS	R/W	0x00	GPIO Interrupt Sense 0: Edge on corresponding pin is detected (edge-sensitive) 1: Level on corresponding pin is detected (level-sensitive)

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

The **GPIOIBE** register is the interrupt both-edges register. When the corresponding bit in the **GPIO Interrupt Sense (GPIOIS)** register (see page 102) 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 104). Clearing a bit configures the pin to be controlled by **GPIOIEV**. All bits are cleared by a reset.

GPIO Interrupt Both Edges (GPIOIBE)

Offset 0x408



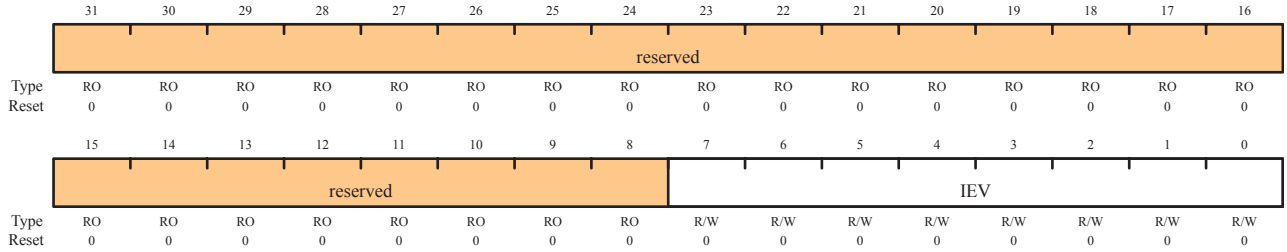
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	IBE	R/W	0x00	GPIO Interrupt Both Edges 0: Interrupt generation is controlled by the **GPIO Interrupt Event (GPIOIEV)** register (see page 142). 1: Both edges on the corresponding pin trigger an interrupt. **Note:** Single edge is determined by the corresponding bit in **GPIOIEV**.

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

The **GPIOIEV** register is the interrupt event register. Bits set to High in **GPIOIEV** configure the corresponding pin to detect rising edges or high levels, depending on the corresponding bit value in the **GPIO Interrupt Sense (GPIOIS)** register (see page 102). 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)

Offset 0x40C



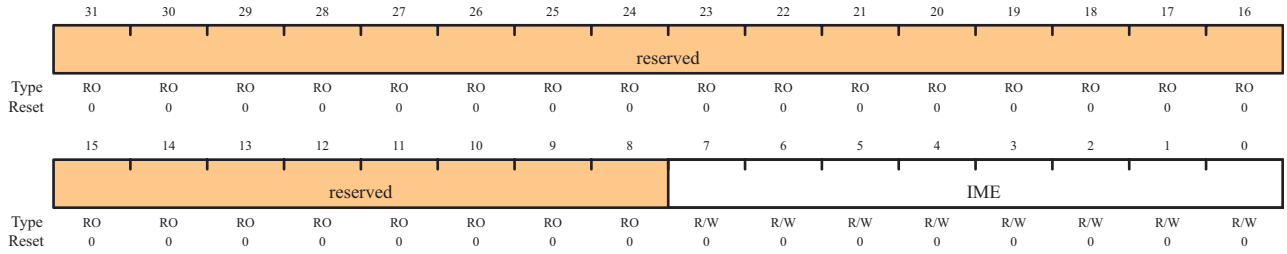
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	IEV	R/W	0x00	GPIO Interrupt Event 0: Falling edge or Low levels on corresponding pins trigger interrupts. 1: Rising edge or High levels on corresponding pins trigger interrupts.

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

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

GPIO Interrupt Mask (GPIOIM)

Offset 0x410



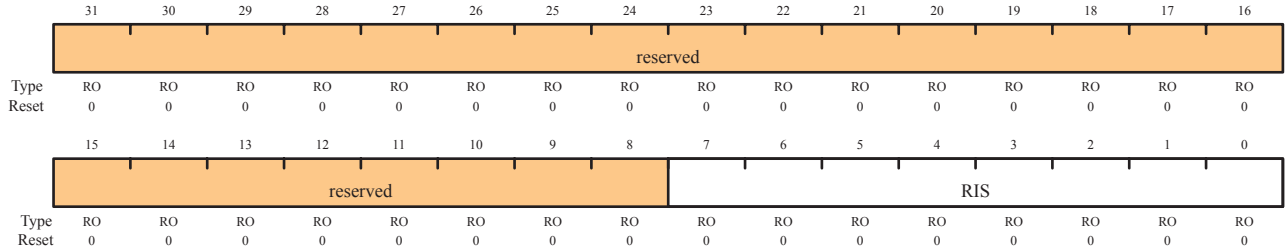
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	IME	R/W	0x00	GPIO Interrupt Mask Enable 0: Corresponding pin interrupt is masked. 1: Corresponding pin interrupt is not masked.

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

The **GPIORIS** register is the raw interrupt status register. Bits read High in **GPIORIS** reflect the status of interrupt trigger conditions detected (raw, prior to masking), indicating that all the requirements have been met, before they are finally allowed to trigger by the **GPIO Interrupt Mask (GPIOIM)** register (see page 105). 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)

Offset 0x414



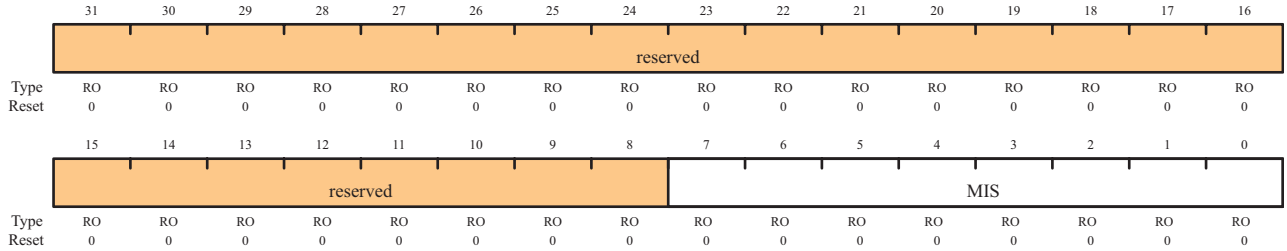
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	RIS	RO	0x00	GPIO Interrupt Raw Status Reflect the status of interrupt trigger condition detection on pins (raw, prior to masking). 0: Corresponding pin interrupt requirements not met. 1: Corresponding pin interrupt has met requirements.

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

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

GPIOMIS is the state of the interrupt after masking.

GPIO Masked Interrupt Status (GPIOMIS)
Offset 0x418



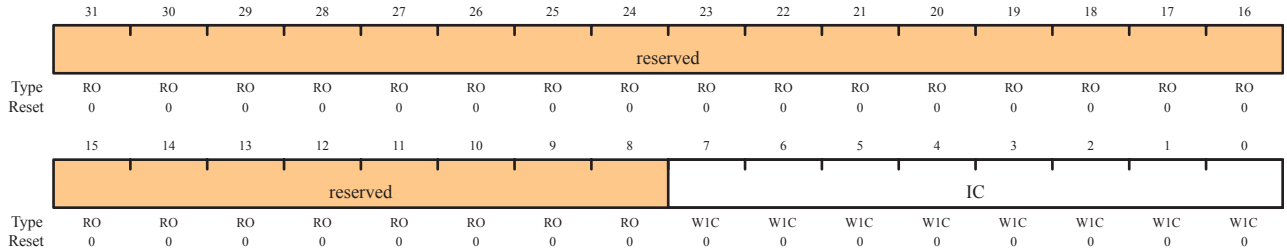
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	MIS	RO	0x00	GPIO Masked Interrupt Status Masked value of interrupt due to corresponding pin. 0: Corresponding GPIO line interrupt not active. 1: Corresponding GPIO line asserting interrupt.

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

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

GPIO Interrupt Clear (GPIOICR)

Offset 0x41C



Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	IC	W1C	0x00	GPIO Interrupt Clear 0: Corresponding interrupt is unaffected. 1: Corresponding interrupt is cleared.

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

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

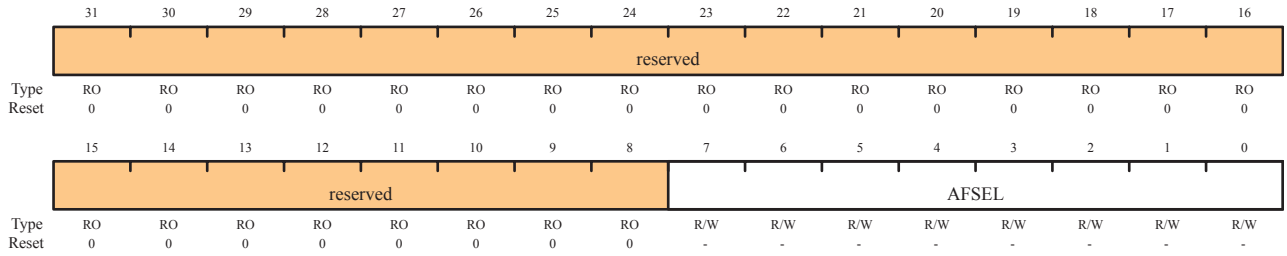
Caution – All GPIO pins are inputs by default (GPIODIR=0 and GPIOAFSEL=0), with the exception of the five JTAG pins (PB7 and PC[3:0]). The JTAG pins default to their JTAG functionality (GPIOAFSEL=1). Asserting a Power-On-Reset (POR) or an external reset (RST) puts both groups of pins back to their default state.

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

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

GPIO Alternate Function Select (GPIOAFSEL)

Offset 0x420



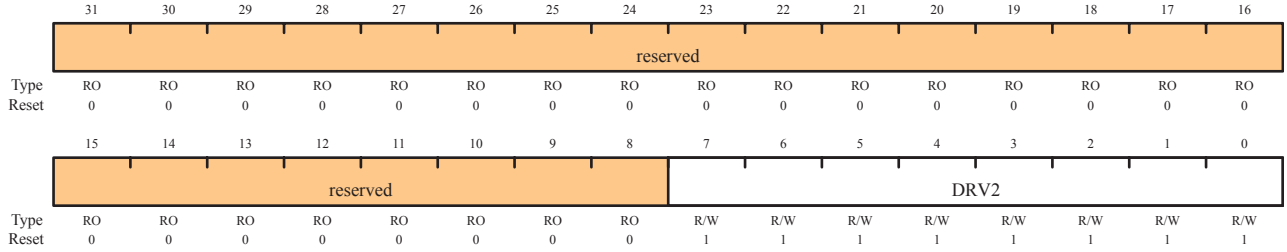
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	AFSEL	R/W	see note	GPIO Alternate Function Select 0: Software control of corresponding GPIO line (GPIO mode). 1: Hardware control of corresponding GPIO line (alternate hardware function). **Note:** The default reset value for the **GPIOAFSEL** register is 0x00 for all GPIO pins, with the exception of the five JTAG pins (PB7 and PC[3:0]). These five pins default to JTAG functionality. Because of this, the default reset value of **GPIOAFSEL** for GPIO Port B is 0x80 while the default reset value of **GPIOAFSEL** for Port C is 0x0F.

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)

Offset 0x500



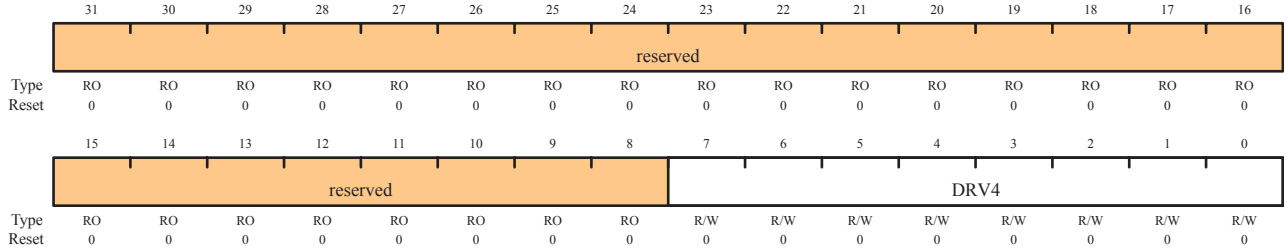
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	DRV2	R/W	0xFF	Output Pad 2-mA Drive Enable A write of 1 to either **GPIODR4[n]** or **GPIODR8[n]** clears the corresponding 2-mA enable bit. The change is effective on the second clock cycle after the write.

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

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

GPIO 4-mA Drive Select (GPIODR4R)

Offset 0x504



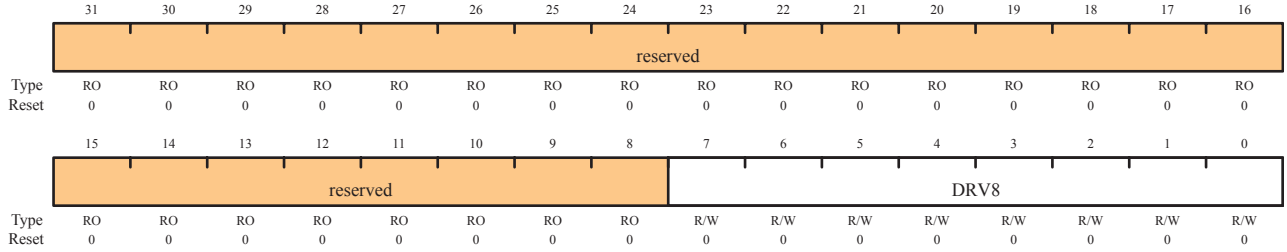
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	DRV4	R/W	0x00	Output Pad 4-mA Drive Enable A write of 1 to either **GPIODR2[n]** or **GPIODR8[n]** clears the corresponding 4-mA enable bit. The change is effective on the second clock cycle after the write.

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

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

GPIO 8-mA Drive Select (GPIODR8R)

Offset 0x508



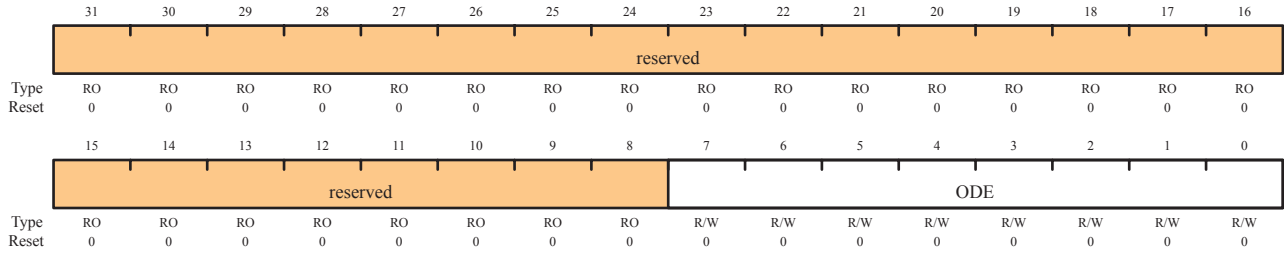
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	DRV8	R/W	0x00	Output Pad 8-mA Drive Enable A write of 1 to either **GPIODR2[n]** or **GPIODR4[n]** clears the corresponding 8-mA enable bit. The change is effective on the second clock cycle after the write.

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

The **GPIOODR** register is the open drain control register. Setting a bit in this register enables the open drain configuration of the corresponding GPIO pad. When the open drain mode is enabled, the corresponding bit should also be set in the **GPIO Digital Input Enable (GPIODEN)** register (see page 117). 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.

GPIO Open Drain Select (GPIOODR)

Offset 0x50C



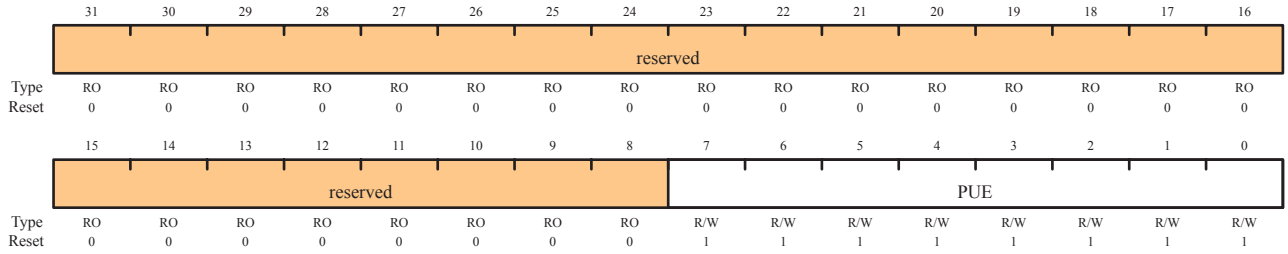
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	ODE	R/W	0x00	Output Pad Open Drain Enable 0: Open drain configuration is disabled. 1: Open drain configuration is enabled.

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

The **GPIOPUR** register is the pull-up control register. When a bit is set to 1, it enables a weak pull-up resistor on the corresponding GPIO signal. Setting a bit in **GPIOPUR** automatically clears the corresponding bit in the **GPIO Pull-Down Select (GPIOPDR)** register (see page 115).

GPIO Pull-Up Select (GPIOPUR)

Offset 0x510



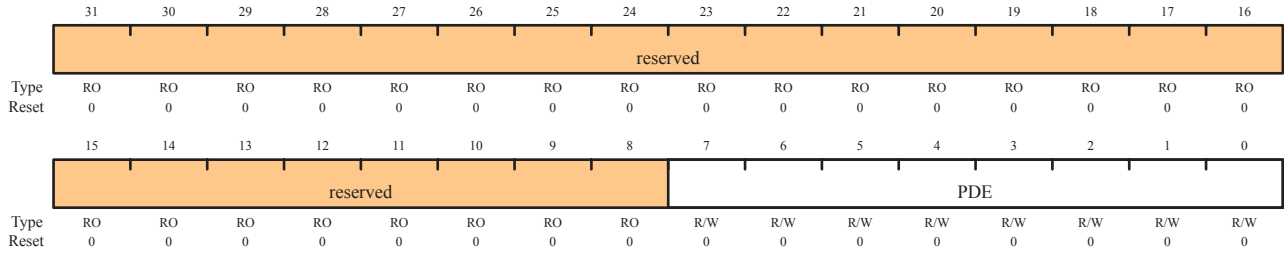
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PUE	R/W	0xFF	Pad Weak Pull-Up Enable A write of 1 to **GPIOPDR[n]** clears the corresponding **GPIOPUR[n]** enables. The change is effective on the second clock cycle after the write.

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

The **GPIOPDR** register is the pull-down control register. When a bit is set to 1, it enables a weak pull-down resistor on the corresponding GPIO signal. Setting a bit in **GPIOPDR** automatically clears the corresponding bit in the **GPIO Pull-Up Select (GPIOPUR)** register (see page 114).

GPIO Pull-Down Select (GPIOPDR)

Offset 0x514



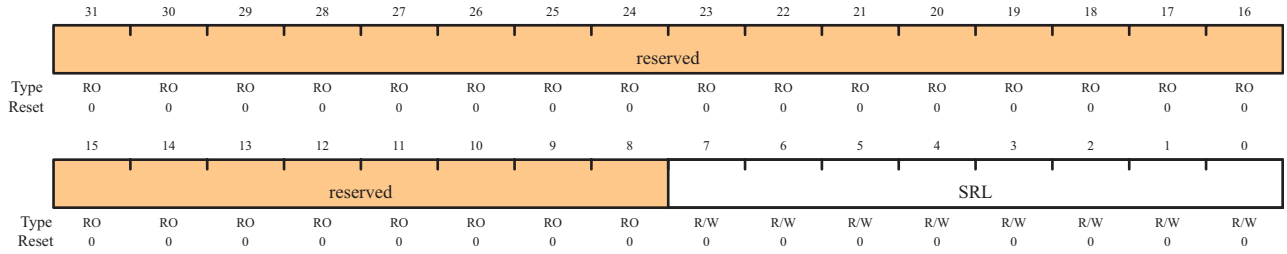
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PDE	R/W	0x00	Pad Weak Pull-Down Enable A write of 1 to **GPIOPUR[n]** clears the corresponding **GPIOPDR[n]** enables. The change is effective on the second clock cycle after 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 112).

GPIO Slew Rate Control Select (GPIOSLR)

Offset 0x518



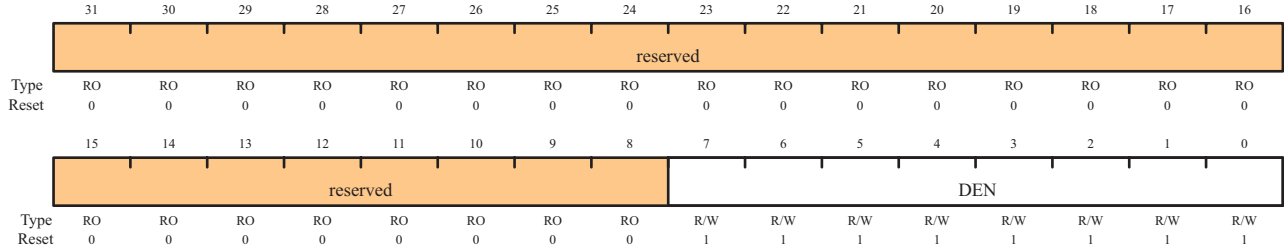
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	SRL	R/W	0	Slew Rate Limit Enable (8-mA drive only) 0: Slew rate control disabled. 1: Slew rate control enabled.

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

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

GPIO Digital Input Enable (GPIODEN)

Offset 0x51C



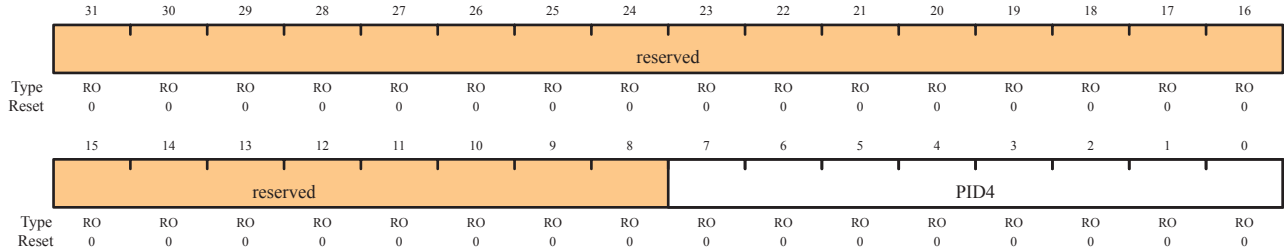
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	DEN	R/W	0xFF	Digital-Input Enable 0: Digital Input disabled 1: Digital Input enabled

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

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

GPIO Peripheral Identification 4 (GPIOPeriphID4)

Offset 0xFD0



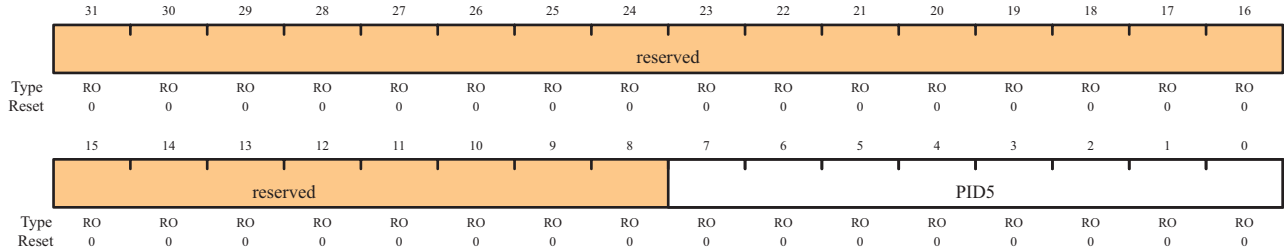
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID4	RO	0x00	GPIO Peripheral ID Register[7:0]

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

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

GPIO Peripheral Identification 5 (GPIOPeriphID5)

Offset 0xFD4



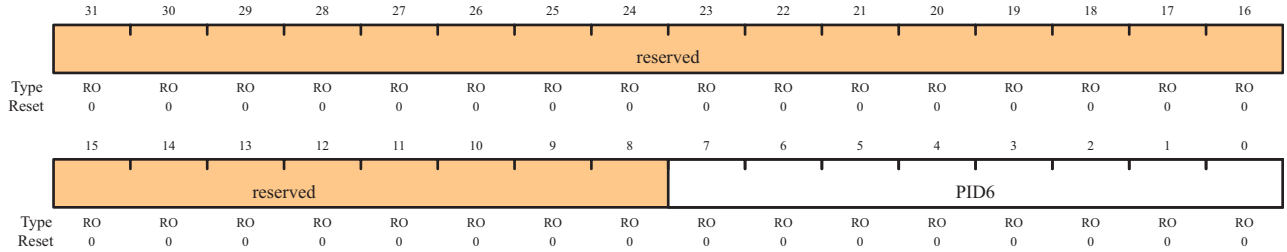
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID5	RO	0x00	GPIO Peripheral ID Register[15:8]

Register 21: GPIO Peripheral Identification 6 (GPIOPeriphID6), offset 0xFD8

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

GPIO Peripheral Identification 6 (GPIOPeriphID6)

Offset 0xFD8



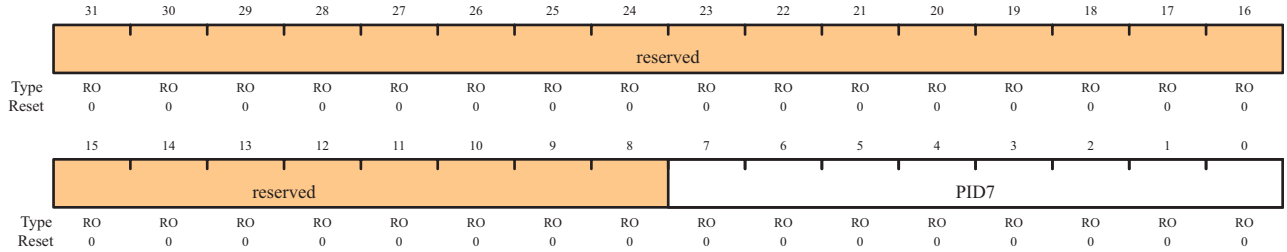
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID6	RO	0x00	GPIO Peripheral ID Register[23:16]

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

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

GPIO Peripheral Identification 7 (GPIOPeriphID7)

Offset 0xFDC



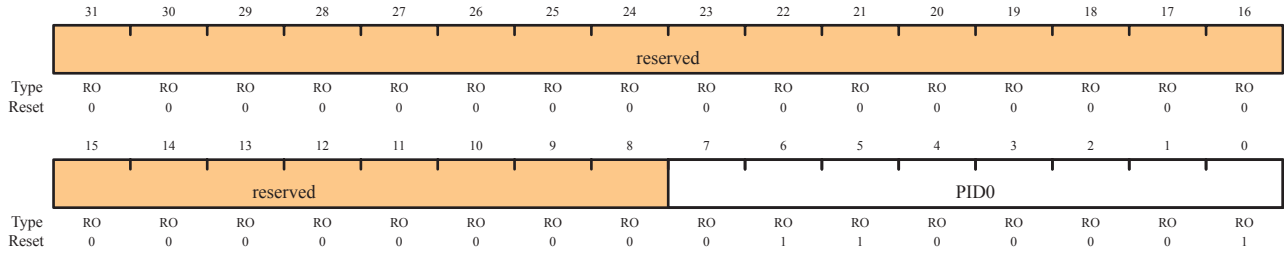
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID7	RO	0x00	GPIO Peripheral ID Register[31:24]

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

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

GPIO Peripheral Identification 0 (GPIOPeriphID0)

Offset 0xFE0



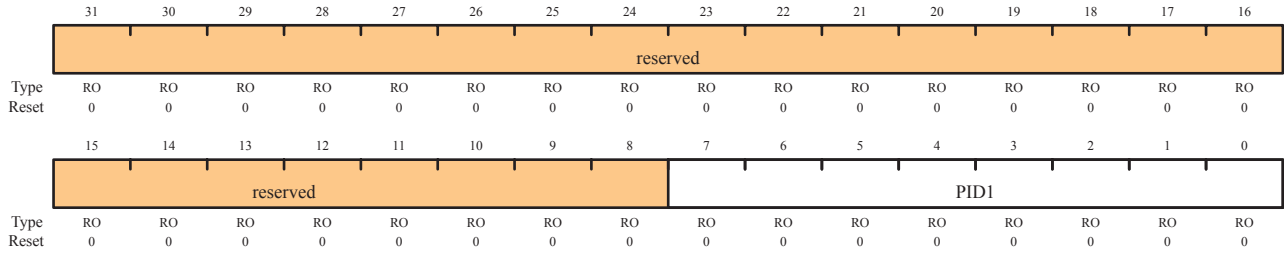
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID0	RO	0x61	GPIO Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 24: GPIO Peripheral Identification 1(GPIOPeriphID1), offset 0xFE4

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

GPIO Peripheral Identification 1 (GPIOPeriphID1)

Offset 0xFE4



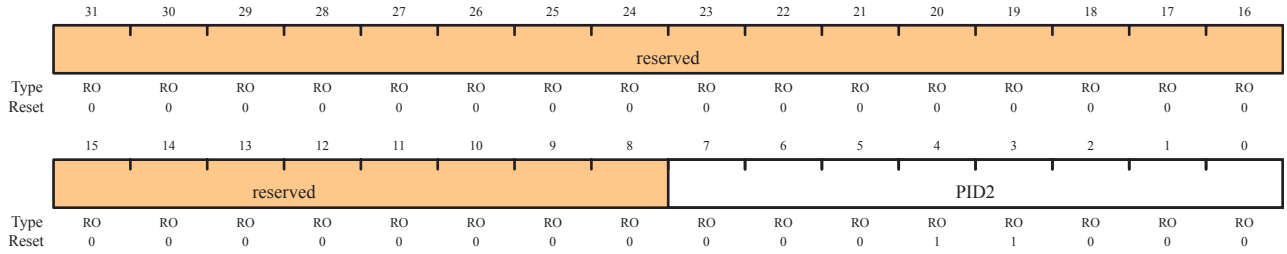
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID1	RO	0x00	GPIO Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

Register 25: GPIO Peripheral Identification 2 (GPIOPeriphID2), offset 0xFE8

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

GPIO Peripheral Identification 2 (GPIOPeriphID2)

Offset 0xFE8



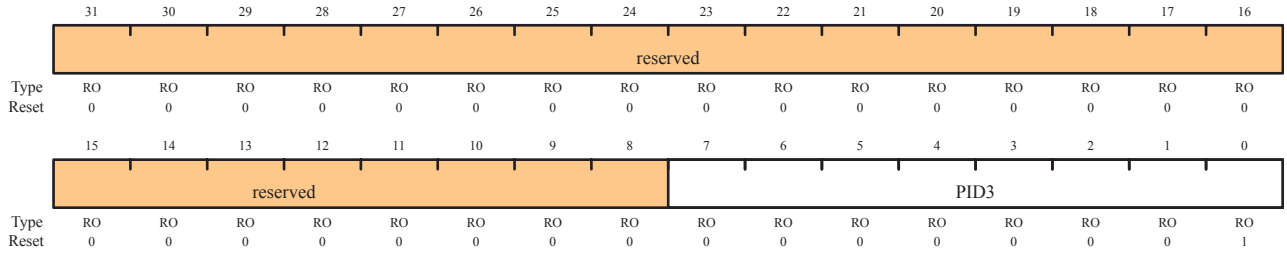
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID2	RO	0x18	GPIO Peripheral ID Register[23:16] Can be used by software to identify the presence of this peripheral.

Register 26: GPIO Peripheral Identification 3 (GPIOPeriphID3), offset 0xFEC

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

GPIO Peripheral Identification 3 (GPIOPeriphID3)

Offset 0xFEC



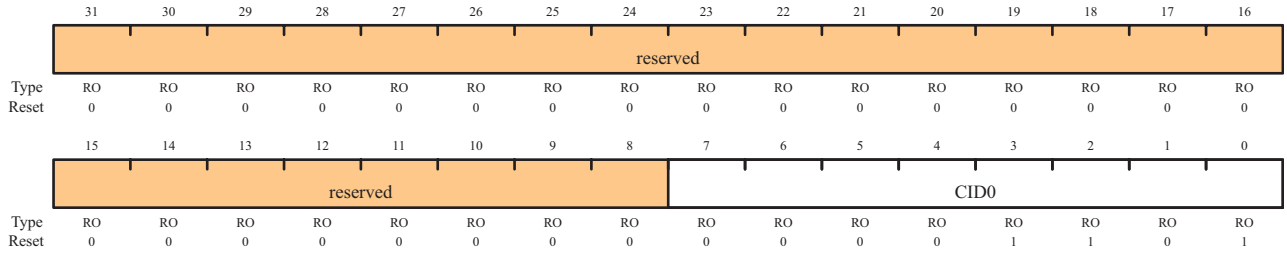
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID3	RO	0x01	GPIO Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

Register 27: GPIO PrimeCell Identification 0 (GPIOPCellID0), offset 0xFF0

The **GPIOPCellID0**, **GPIOPCellID1**, **GPIOPCellID2**, and **GPIOPCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIO Primecell Identification 0 (GPIOPCellID0)

Offset 0xFF0



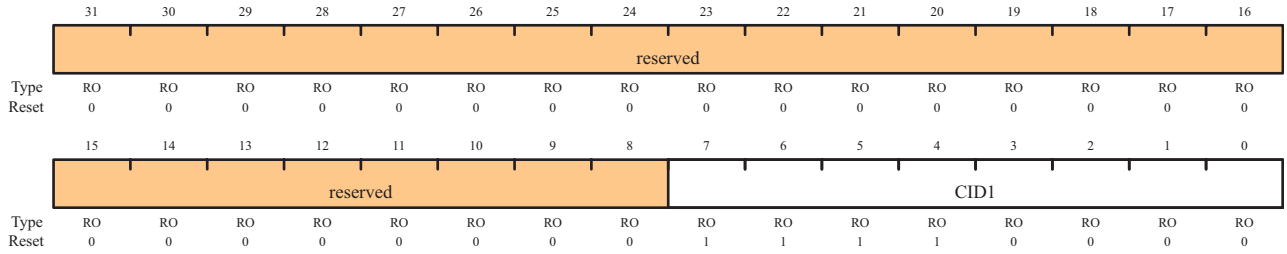
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CID0	RO	0x0D	GPIO PrimeCell ID Register[7:0] Provides software a standard cross-peripheral identification system.

Register 28: GPIO PrimeCell Identification 1 (GPIOPCellID1), offset 0xFF4

The **GPIOPCellID0**, **GPIOPCellID1**, **GPIOPCellID2**, and **GPIOPCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIO Primecell Identification 1 (GPIOPCellID1)

Offset 0xFF4



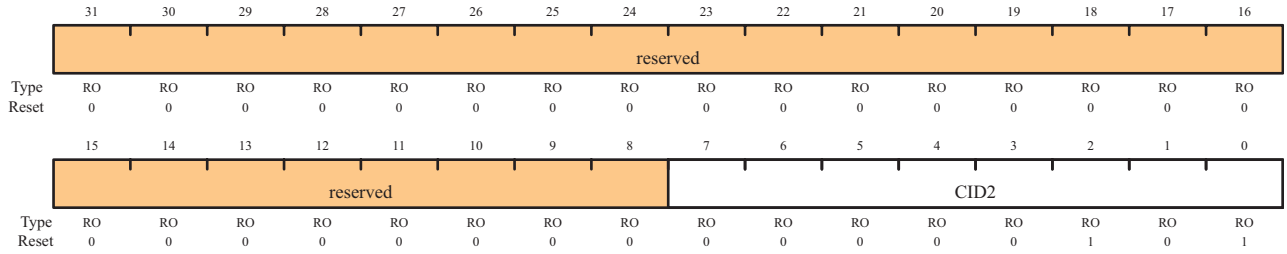
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CID1	RO	0xF0	GPIO PrimeCell ID Register[15:8] Provides software a standard cross-peripheral identification system.

Register 29: GPIO PrimeCell Identification 2 (GPIOPCellID2), offset 0xFF8

The **GPIOPCellID0**, **GPIOPCellID1**, **GPIOPCellID2**, and **GPIOPCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIO Primecell Identification 2 (GPIOPCellID2)

Offset 0xFF8



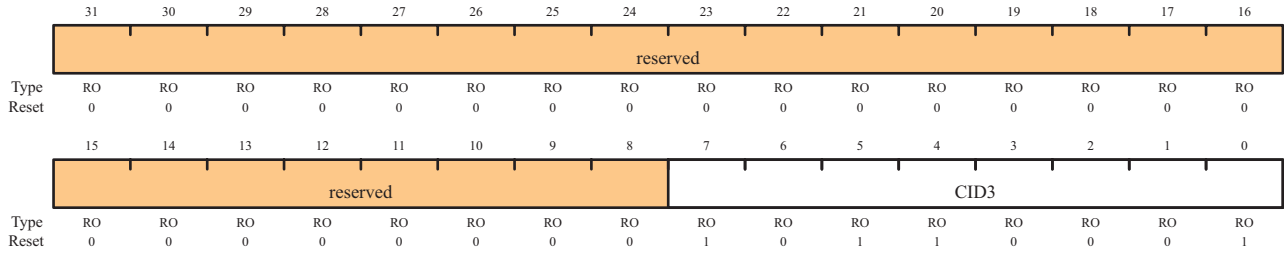
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CID2	RO	0x05	GPIO PrimeCell ID Register[23:16] Provides software a standard cross-peripheral identification system.

Register 30: GPIO PrimeCell Identification 3 (GPIOPCellID3), offset 0xFFC

The **GPIOPCellID0**, **GPIOPCellID1**, **GPIOPCellID2**, and **GPIOPCellID3** registers are four 8-bit wide registers, that can conceptually be treated as one 32-bit register. The register is used as a standard cross-peripheral identification system.

GPIO Primecell Identification 3 (GPIOPCellID3)

Offset 0xFFC



Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CID3	RO	0xB1	GPIO PrimeCell ID Register[31:24] Provides software a standard cross-peripheral identification system.

9 General-Purpose Timers

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

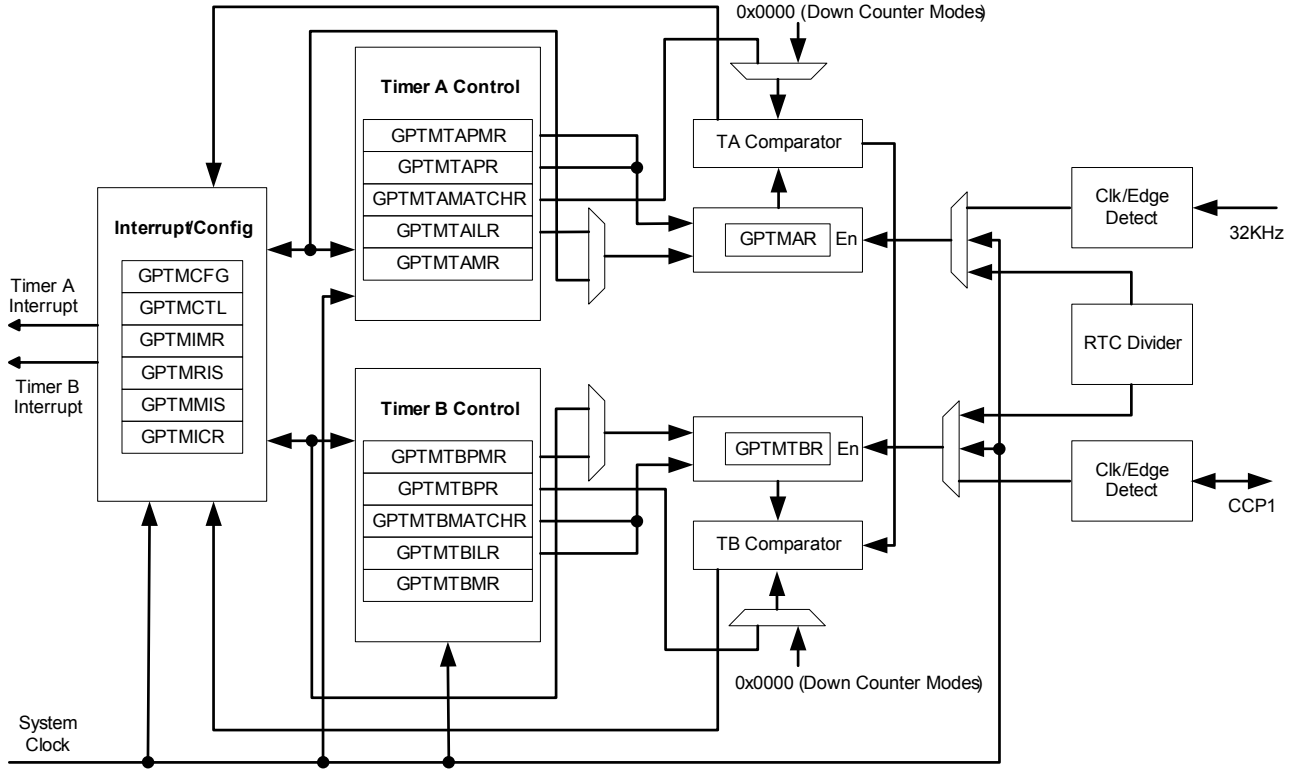
The LM3S101 controller General-Purpose Timer Module (GPTM) contains two GPTM blocks (Timer0 and Timer1). Each GPTM block provides two 16-bit timer/counters (referred to as TimerA and TimerB) that can be configured to operate independently as timers or event counters, or configured to operate as one 32-bit timer or one 32-bit Real-Time Clock (RTC).

The following modes are supported:

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

9.1 Block Diagram

Figure 9-1. GPTM Block Diagram



9.2 Functional Description

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

Software configures the GPTM using the **GPTM Configuration (GPTMCFG)** register (see page 141), the **TimerA Mode (GPTMTAMR)** register (see page 142), and the **TimerB Mode (GPTMTBMR)** register (see page 143). When in one of the 32-bit modes, the timer can only act as a 32-bit timer. However, when configured in 16-bit mode, the GPTM can have its two 16-bit timers configured in any combination of the 16-bit modes.

9.2.1 GPTM Reset Conditions

After reset has been applied to the GPTM module, the module is in an inactive state, and all control registers are cleared and in their default states. Counters TimerA and TimerB are initialized to 0xFFFF, along with their corresponding load registers: the **GPTM TimerA Interval Load (GPTMTAILR)** register (see page 150) and the **GPTM TimerB Interval Load (GPTMTBILR)** register (see page 151). The prescale counters are initialized to 0x00: the **GPTM TimerA Prescale (GPTMTAPR)** register (see page 154) and the **GPTM TimerB Prescale (GPTMTBPR)** register (see page 155).

9.2.2 32-Bit Timer Operating Modes

Note: The odd-numbered CCP pins are used for 16-bit input and the even-numbered CCP pins are used for 32-bit input.

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 150
- **GPTM TimerB Interval Load (GPTMTBILR)** register [15:0], see page 151
- **GPTM TimerA (GPTMTAR)** register [15:0], see page 158
- **GPTM TimerB (GPTMTBR)** register [15:0], see page 159

In the 32-bit modes, the GPTM translates a 32-bit write access to **GPTMTAILR** into a write access to both **GPTMTAILR** and **GPTMTBILR**. The resulting word ordering for such a write operation is: `GPTMTBILR[15:0]:GPTMTAILR[15:0]`. Likewise, a read access to **GPTMTAR** returns the value: `GPTMTBR[15:0]:GPTMTAR[15:0]`.

9.2.2.1 32-Bit One-Shot/Periodic Timer Mode

In 32-bit one-shot and periodic timer modes, the concatenated versions of the TimerA and TimerB registers are configured as a 32-bit down-counter. The selection of one-shot or periodic mode is determined by the value written to the `TAMR` field of the **GPTM TimerA Mode (GPTMTAMR)** register (see page 142), 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 144), the timer begins counting down from its preloaded value. Once the 0x00000000 state is reached, the timer reloads its start value from the concatenated **GPTMTAILR** on the next cycle. If configured to be a one-shot timer, the timer stops counting and clears the `TAEN` bit in the **GPTMCTL** register. If configured as a periodic timer, it continues counting.

In addition to reloading the count value, the GPTM generates interrupts and output triggers when it reaches the 0x0000000 state. The GPTM sets the `TATORIS` bit in the **GPTM Raw Interrupt Status (GPTMRIS)** register (see page 147), and holds it until it is cleared by writing the **GPTM Interrupt Clear (GPTMICR)** register (see page 149). If the time-out interrupt is enabled in the **GPTM Interrupt Mask (GPTIMR)** register (see page 146), the GPTM also sets the `TATOMIS` bit in the **GPTM Masked Interrupt Status (GPTMISR)** register (see page 148).

The output trigger is a one-clock-cycle pulse that is asserted when the counter hits the 0x00000000 state, and deasserted on the following clock cycle. It is enabled by setting the `TAOTE` bit in **GPTMCTL**.

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

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

9.2.2.2 32-Bit Real-Time Clock Timer Mode

In Real-Time Clock (RTC) mode, the concatenated versions of the TimerA and TimerB registers are configured as a 32-bit up-counter. When RTC mode is selected for the first time, the counter is loaded with a value of 0x00000001. All subsequent load values must be written to the **GPTM TimerA Match (GPTMTAMATCHR)** register (see page 152) by the controller.

The 32KHZ pin is dedicated to the 32-bit RTC function, and the input clock is 32 KHz.

When software writes the TAEN bit in **GPTMCTL**, the counter starts counting up from its preloaded value of 0x00000001. When the current count value matches the preloaded value in **GPTMTAMATCHR**, it rolls over to a value of 0x00000000 and continues counting until either a hardware reset, or it is disabled by software (clearing the TAEN bit). When a match occurs, the GPTM asserts the RTCRIS bit in **GPTMRIS**. If the RTC interrupt is enabled in **GPTIMR**, the GPTM also sets the RTCMIS bit in **GPTMISR** and generates a controller interrupt. The status flags are cleared by writing the RTCCINT bit in **GPTMICR**.

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

9.2.3 16-Bit Timer Operating Modes

The GPTM is placed into global 16-bit mode by writing a value of 0x4 to the **GPTM Configuration (GPTMCFG)** register (see page 141). This section describes each of the GPTM 16-bit modes of operation. Timer A and Timer B have identical modes, so a single description is given using an *n* to reference both.

9.2.3.1 16-Bit One-Shot/Periodic Timer Mode

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

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

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

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

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

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

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

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

Prescale	#Clock (T_c)[a]	Max Time	Units
00000000	1	1.3107	mS
00000001	2	2.6214	mS

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

Prescale	#Clock (T_c)[a]	Max Time	Units
00000010	3	3.9321	mS
----------	--		
11111100	254	332.9229	mS
11111110	255	334.2336	mS
11111111	256	335.5443	mS

a. T_c is the clock period.

9.2.3.2 16-Bit Input Edge Count Mode

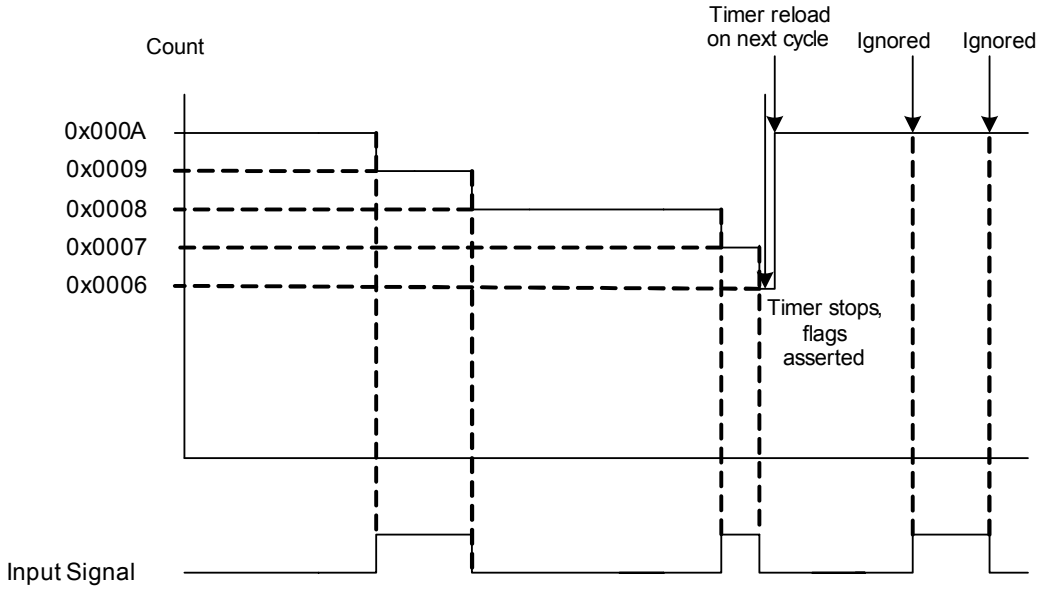
In Edge Count mode, the timer is configured as a down-counter capable of capturing three types of events: rising edge, falling edge, or both. To place the timer in Edge Count mode, the TnCMR bit of the **GPTMTnMR** register must be set to 0. The type of edge that the timer counts is determined by the TnEVENT fields of the **GPTMCTL** register. During initialization, the **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 (and the CnMMIS bit, if the interrupt is not masked). The counter is then reloaded using the value in **GPTMTnILR**, and stopped since the GPTM automatically clears the TnEN bit in the **GPTMCTL** register. Once the event count has been reached, all further events are ignored until TnEN is re-enabled by software.

Figure 9-2 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 4 edge events are counted. The counter is configured to detect both edges of the input signal.

Note that the last two edges are not counted since the timer automatically clears the TnEN bit after the current count matches the value in the **GPTMnMR** register.

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



9.2.3.3 16-Bit Input Edge Time Mode

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

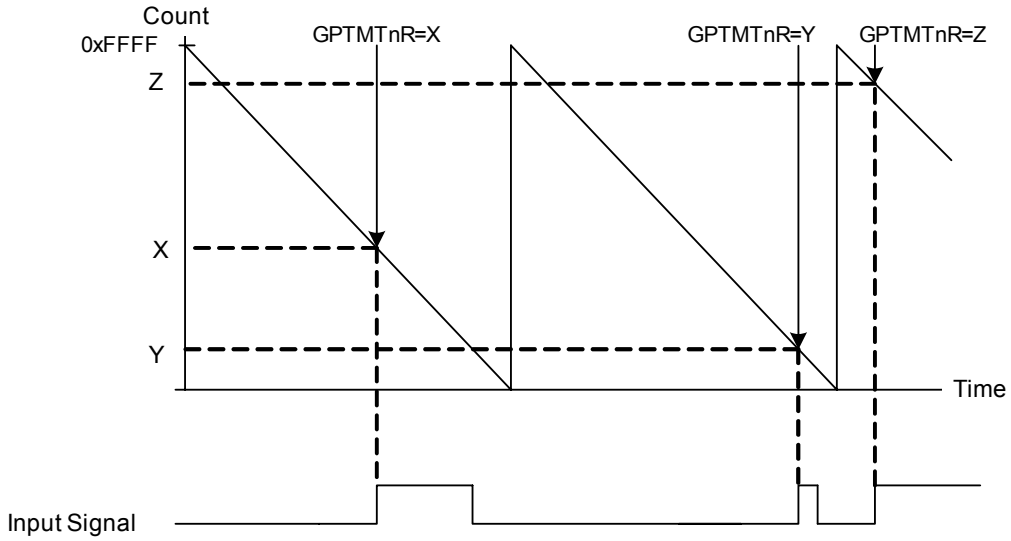
When software writes the `TnEN` bit in the **GPTMCTL** register, the timer is enabled for event capture. When the selected input event is detected, the current Tn counter value is captured in the **GPTMTnR** register and is available to be read by the controller. The GPTM then asserts the `CnERIS` bit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

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

Figure 9-3 shows how input edge timing mode works. In the diagram, it is assumed that the start value of the timer is the default value of 0xFFFF, and the timer is configured to capture rising edge events.

Each time a rising edge event is detected, the current count value is loaded into the **GPTMTnR** register, and is held there until another rising edge is detected (at which point the new count value is loaded into **GPTMTnR**).

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



9.2.3.4 16-Bit PWM Mode

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

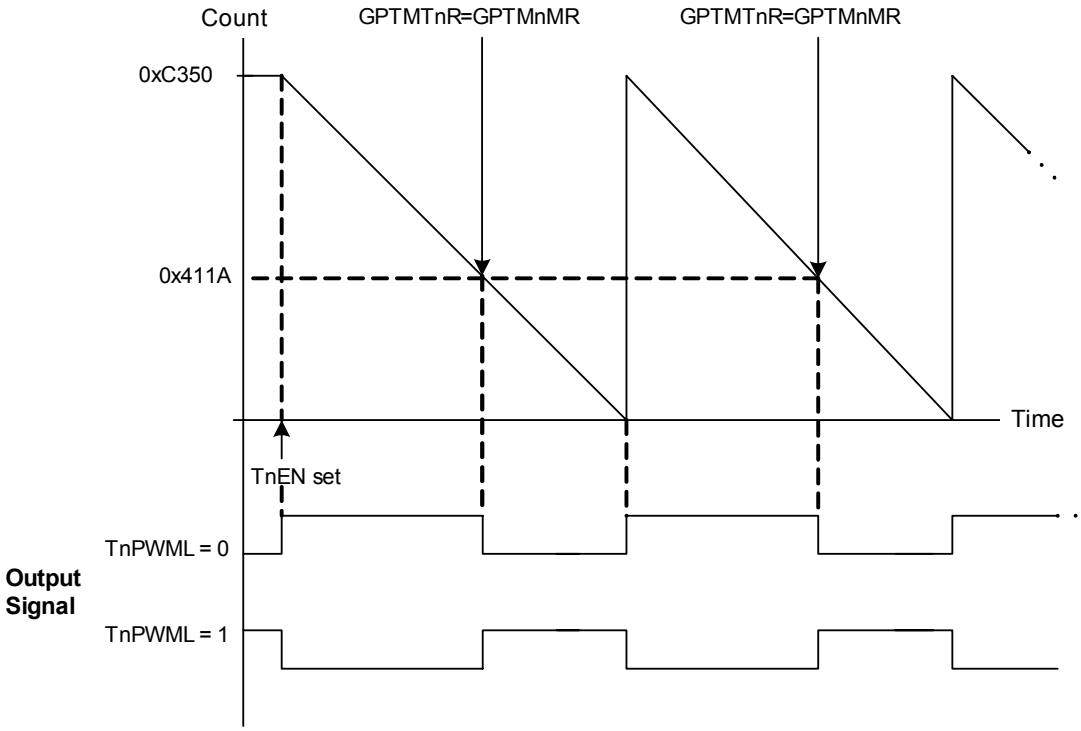
PWM mode can take advantage of the 8-bit prescaler by using the **Timern Prescale Register (GPTMTnPR)** and the **Timern Prescale Match Register (GPTMTnPMR)**. This effectively extends the range of the timer to 24 bits.

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

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

Figure 9-4 shows how to generate an output PWM with a 1-ms period and a 66% duty cycle assuming a 50 MHz input clock and **TnPWML**=0 (duty cycle would be 33% for the **TnPWML**=1 configuration). For this example, the start value is **GPTMnIRL**=0xC350 and the match value is **GPTMnMR**=0x411A.

Figure 9-4. 16-Bit PWM Mode Example



9.3 Initialization and Configuration

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

9.3.1 32-bit One-Shot/Periodic Timer Mode

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

1. Ensure the timer is disabled (the TAEN bit in the **GPTMCTL** register is cleared) before making any changes.
2. Write the **Configuration Register (GPTMCFG)** to a value of 0x0.
3. Set the TAMR field in the **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 **TimerA Interval Load Register (GPTMTAILR)**.
5. If interrupts are required, set the TATOIM bit in the **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 **Interrupt Clear Register (GPTMICR)**.

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

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

To use the RTC mode, the timer must have a 32-KHz input signal on its 32KHz pin. 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 **Configuration Register (GPTMCFG)** with a value of 0x1.
3. Write the desired match value to the **TimerA Match Register (GPTMTAMATCHR)**.
4. Set/clear the RTCEN bit in the **Control Register (GPTMCTL)** as desired.
5. If interrupts are required, set the RTCIM bit in the **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 0x00000000 and begins counting. If an interrupt is enabled, it does not have to be cleared.

9.3.3 16-Bit One-Shot/Periodic Timer Mode

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

1. Ensure the target timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **Configuration Register (GPTMCFG)** to a value of 0x4.
3. Set the TnMR field in the **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 **Timern Prescale Register (GPTMTnPR)**.
5. Load the start value into the **Timer Interval Load Register (GPTMTnILR)**.
6. If interrupts are required, set the TnTOIM bit in the **Interrupt Mask Register (GPTMIMR)**.
7. Set the TnEN bit in the **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 **Interrupt Clear Register (GPTMICR)**.

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

9.3.4 16-Bit Input Edge Count Mode

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

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register to 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 will capture by writing the TnEVENT field of the **GPTM Control (GPTMCTL)** register.
5. Load the timer start value into the **Timern Interval Load (GPTMTnILR)** register.
6. Load the desired event count into the **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 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 **Interrupt Clear (GPTMICR)** register.

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

9.3.5 16-Bit Input Edge Timing Mode

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

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register to 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 will capture by writing the TnEVENT field of the **GPTM Control (GPTMCTL)** register.
5. Load the timer start value into the **Timern Interval Load (GPTMTnILR)** register.
6. If interrupts are required, set the CnEIM bit in the **GPTM Interrupt Mask (GPTMIMR)** register.
7. Set the TnEN bit in the **GPTM Control (GPTMCTL)** register to enable the timer and start counting.
8. Poll the CnERIS bit in the **GPTMRIS** register or wait for the interrupt to be generated (if enabled). In both cases, the status flags are cleared by writing a 1 to the CnECINT bit of the **GPTM Interrupt Clear (GPTMICR)** register. The time at which the event happened can be obtained by reading the **GPTM Timern (GPTMTnR)** register.

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

9.3.6 16-Bit PWM Mode

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

1. Ensure the timer is disabled (the TnEN bit is cleared) before making any changes.
2. Write the **GPTM Configuration (GPTMCFG)** register to a value of 0x4.
3. In the **GPTM Timer Mode (GPTMTnMR)** register, set the TnAMS field to 0x1 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 **Timern Interval Load (GPTMTnILR)** register.
6. Load the **Timern Match (GPTMTnMATCHR)** register with the desired value.
7. If a prescaler is going to be used, configure the **Timern Prescale (GPTMTnPR)** register and the **Timern Prescale Match (GPTMTnPMR)** register.
8. Set the TnEN bit in the **GPTM Control (GPTMCTL)** register to enable the timer and begin generation of the output PWM signal.

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

9.4 Register Map

Table 9-1 lists the GPTM registers. All addresses given are relative to that timer's base address:

- Timer0: 0x40030000
- Timer1: 0x40031000

Table 9-2. GPTM Register Map

Offset	Name	Reset	Type	Description	See page
0x000	GPTMCFG	0x00000000	R/W	Configuration reset value	141
0x004	GPTMTAMR	0x00000000	R/W	TimerA mode reset value	142
0x008	GPTMTBMR	0x00000000	R/W	TimerB mode reset value	143
0x00C	GPTMCTL	0x00000000	R/W	Control reset value	144
0x018	GPTMIMR	0x00000000	R/W	Interrupt mask reset value	146
0x01C	GPTMRIS	0x00000000	RO	Interrupt status reset value	147
0x020	GPTMMIS	0x00000000	RO	Masked interrupt status reset value	148
0x024	GPTMICR	0x00000000	W1C	Interrupt clear reset value	149
0x028	GPTMTAILR	-	R/W	TimerA interval load reset value	150
0x02C	GPTMTBILR	0x0000FFFF	R/W	TimerB interval load reset value	151
0x030	GPTMTAMATCHR	-	R/W	TimerA match reset value	152
0x034	GPTMTBMATCHR	0x0000FFFF	R/W	TimerB match reset value	153
0x038	GPTMTAPR	0x00000000	R/W	TimerA prescale reset value	154
0x03C	GPTMTBPR	0x00000000	R/W	TimerB prescale reset value	155
0x040	GPTMTAPMR	0x00000000	R/W	TimerA prescale match reset value	156
0x044	GPTMTBPMR	0x00000000	R/W	TimerB prescale match reset value	157
0x048	GPTMTAR	0xFFFFFFFF	RO	TimerA reset value	158
0x04C	GPTMTBR	0x0000FFFF	RO	TimerB reset value	159

9.5 Register Descriptions

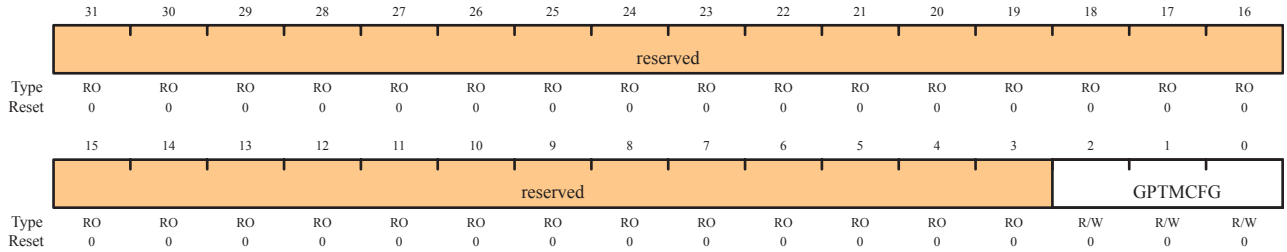
The remainder of this section lists and describes the GPTM registers, in numerical order by address offset.

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

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

GPTM Configuration (GPTMCFG)

Offset 0x000



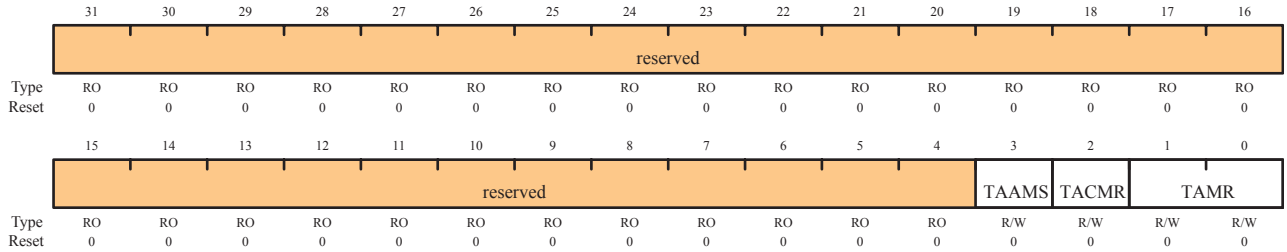
Bit	Name	Type	Reset	Description
31:3	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
2:0	GPTMCFG	R/W	0	GPTM Configuration 0x0: 32-bit timer configuration. 0x1: 32-bit real-time clock (RTC) counter configuration. 0x2: Reserved. 0x3: Reserved. 0x4-0x7: 16-bit timer configuration, function is controlled by bits 1:0 of **GPTMTAMR** and **GPTMTBMR**.

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

This register configures the GPTM based on the configuration selected in the GPTMCFG. When in 16-bit PWM mode, the TATMR field should be set to 0x3, and the TACMR should be set to 0x0.

GPTM TimerA Mode (GPTMTAMR)

Offset 0x004



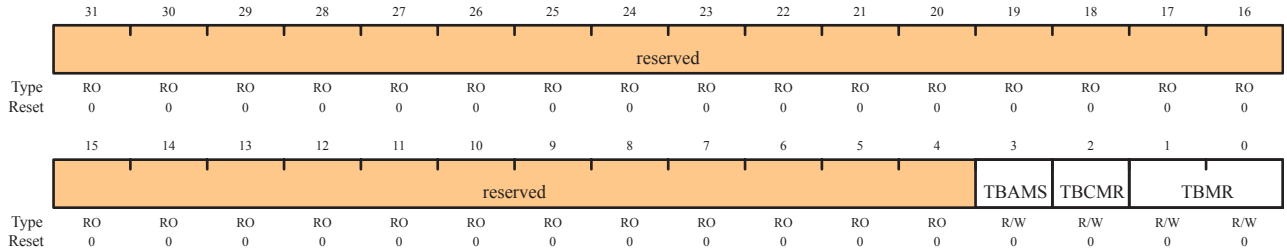
Bit	Name	Type	Reset	Description
31:4	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
3	TAAMS	R/W	0	GPTM TimerA Alternate Mode Select 0: Capture mode is enabled. 1: PWM mode is enabled.
2	TACMR	R/W	0	GPTM TimerA Capture Mode 0: Edge-Count mode. 1: Edge-Time mode.
1:0	TAMR	R/W	0	GPTM TimerA Mode 0x0: Reserved. 0x1: One-Shot Timer mode. 0x2: Periodic Timer mode. 0x3: Capture mode. The Timer mode is based on the timer configuration defined by bits 2:0 in the **GPTMCFG** register (16-or 32-bit). In 16-bit timer configuration, these bits control 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 GPTMCFG. When in 16-bit PWM mode, the TBTMR field should be set to 0x3, and the TBCMR should be set to 0x0.

GPTM TimerB Mode (GPTMTBMR)

Offset 0x008

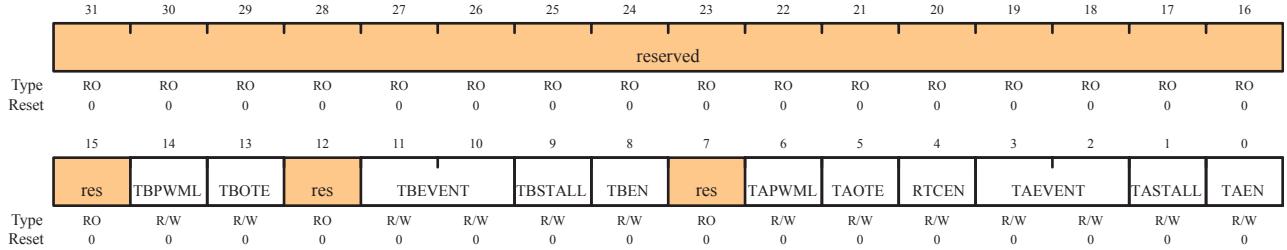


Bit	Name	Type	Reset	Description
31:4	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
3	TBAMS	R/W	0	GPTM TimerB Alternate Mode Select 0: Capture mode is enabled. 1: PWM mode is enabled.
2	TBCMR	R/W	0	GPTM TimerB Capture Mode 0: Edge-Count mode. 1: Edge-Time mode.
1:0	TBMR	R/W	0	GPTM TimerB Mode 0x0: Reserved. 0x1: One-Shot Timer mode. 0x2: Periodic Timer mode. 0x3: Capture mode. The timer mode is based on the timer configuration defined by bits 2:0 in the **GPTMCFG** register. In 16-bit timer configuration, these bits control the 16-bit timer modes for TimerB. In 32-bit timer configuration, this register's contents are ignored and **GPTMTAMR** is used.

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

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

GPTM Control (GPTMCTL)
Offset 0x00C



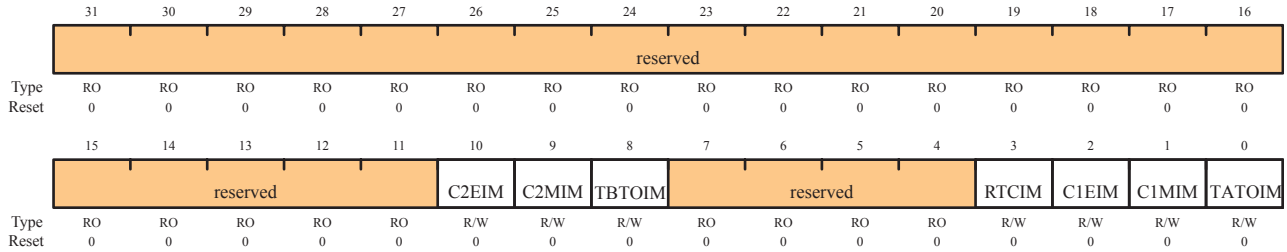
Bit	Name	Type	Reset	Description
31:15	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
14	TBPWML	R/W	0	GPTM TimerB PWM Output Level 0: Output is unaffected. 1: Output is inverted.
13	TBOTE	R/W	0	GPTM TimerB Output Trigger Enable 0: The output TimerB trigger is disabled. 1: The output TimerB trigger is enabled.
12	reserved	RO	0	Read as 0.
11:10	TBEVENT	R/W	0	GPTM TimerB Event Mode 00: Positive Edge. 01: Negative Edge. 11: Both Edges.
9	TBSTALL	R/W	0	GPTM TimerB Stall Enable 0: TimerB stalling is disabled. 1: TimerB stalling is enabled.
8	TBEN	R/W	0	GPTM TimerB Enable 0: TimerB is disabled. 1: TimerB is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register.
7	reserved	RO	0	Read as 0.
6	TAPWML	R/W	0	GPTM TimerA PWM Output Level 0: Output is unaffected. 1: Output is inverted.

Bit	Name	Type	Reset	Description
5	TAOTE	R/W	0	GPTM TimerA Output Trigger Enable 0: The output TimerA trigger is disabled. 1: The output TimerA trigger is enabled.
4	RTCEN	R/W	0	GPTM RTC Enable 0: RTC counting is disabled 1: RTC counting is enabled
3:2	TAEVENT	R/W	0	GPTM TimerA Event Mode 00: Positive edge. 01: Negative edge. 11: Both edges.
1	TASTALL	R/W	0	GPTM TimerA Stall Enable 0: TimerA stalling is disabled. 1: TimerA stalling is enabled.
0	TAEN	R/W	0	GPTM TimerA Enable 0: TimerA is disabled. 1: TimerA is enabled and begins counting or the capture logic is enabled based on the **GPTMCFG** register.

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

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

GPTM Interrupt Mask (GPTMIMR)
Offset 0x018

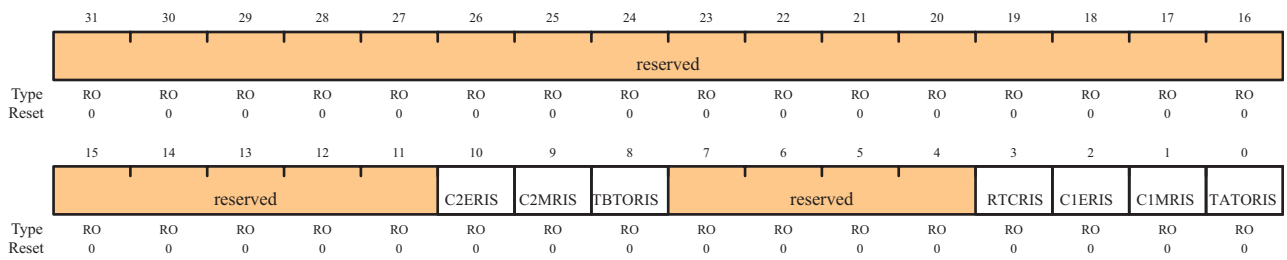


Bit	Name	Type	Reset	Description
31:11	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
10	C2EIM	R/W	0	GPTM Capture2 Event Interrupt Mask 0: Interrupt is disabled. 1: Interrupt is enabled.
9	C2MIM	R/W	0	GPTM Capture2 Match Interrupt Mask 0: Interrupt is disabled. 1: Interrupt is enabled.
8	TBTOIM	R/W	0	GPTM TimerB Time-Out Interrupt Mask 0: Interrupt is disabled. 1: Interrupt is enabled.
7:4	reserved	RO	0	Read as 0.
3	RTCIM	R/W	0	GPTM RTC Interrupt Mask 0: Interrupt is disabled. 1: Interrupt is enabled.
2	C1EIM	R/W	0	GPTM Capture1 Event Interrupt Mask 0: Interrupt is disabled. 1: Interrupt is enabled.
1	C1MIM	R/W	0	GPTM Capture1 Match Interrupt Mask 0: Interrupt is disabled. 1: Interrupt is enabled.
0	TATOIM	R/W	0	GPTM TimerA Time-Out Interrupt Mask 0: Interrupt is disabled. 1: Interrupt is enabled.

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**.

Offset 0x01C



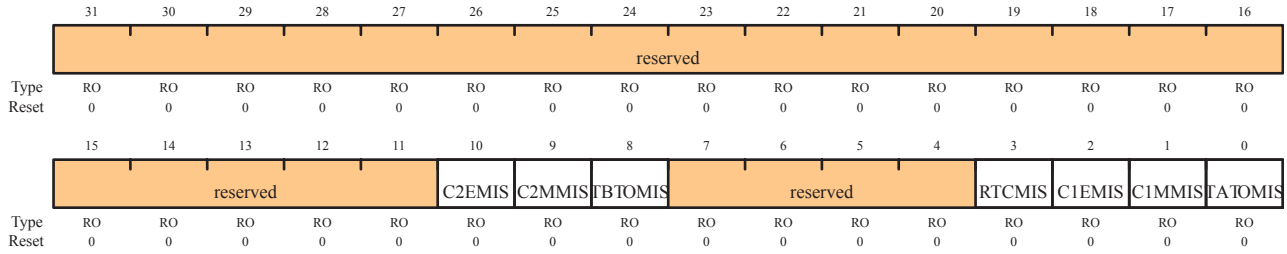
Bit	Name	Type	Reset	Description
31:11	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
10	C2ERIS	RO	0	GPTM Capture2 Event Raw Interrupt This is the Capture2 Event interrupt status prior to masking.
9	C2MRIS	RO	0	GPTM Capture2 Match Raw Interrupt This is the Capture2 Match interrupt status prior to masking.
8	TBTORIS	RO	0	GPTM TimerB Time-Out Raw Interrupt This is the TimerB time-out interrupt status prior to masking.
7:4	reserved	RO	0	Read as 0.
3	RTCRIS	RO	0	GPTM RTC Raw Interrupt This is the RTC Event interrupt status prior to masking.
2	C1ERIS	RO	0	GPTM Capture1 Event Raw Interrupt This is the Capture1 Event interrupt status prior to masking.
1	C1MRIS	RO	0	GPTM Capture1 Match Raw Interrupt This is the Capture1 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)

Offset 0x020



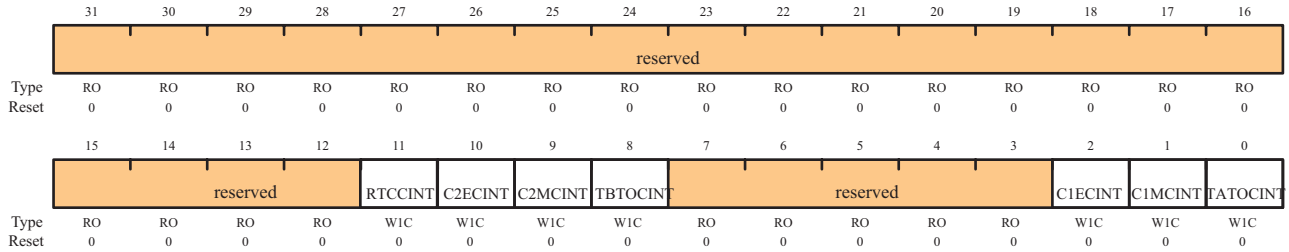
Bit	Name	Type	Reset	Description
31:11	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
10	C2EMIS	RO	0	GPTM Capture2 Event Masked Interrupt This is the Capture2 Event interrupt status after masking.
9	C2MMIS	RO	0	GPTM Capture2 Match Masked Interrupt This is the Capture2 Match interrupt status after masking.
8	TBTOMIS	RO	0	GPTM TimerB Time-Out Masked Interrupt This is the TimerB time-out interrupt status after masking.
7:4	reserved	RO	0	Read as 0s.
3	RTCMIS	RO	0	GPTM RTC Masked Interrupt This is the RTC Event interrupt status after masking.
2	C1EMIS	RO	0	GPTM Capture1 Event Masked Interrupt This is the Capture1 Event interrupt status after masking.
1	C1MMIS	RO	0	GPTM Capture1 Match Masked Interrupt This is the Capture1 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)

Offset 0x024

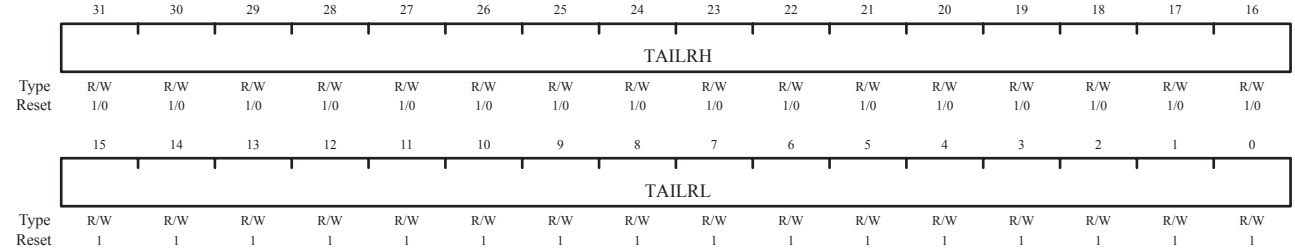


Bit	Name	Type	Reset	Description
31:11	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
10	C2ECINT	W1C	0	GPTM Capture2 Event Interrupt Clear 0: The interrupt is unaffected. 1: The interrupt is cleared.
9	C2MCINT	W1C	0	GPTM Capture2 Match Interrupt Clear 0: The interrupt is unaffected. 1: The interrupt is cleared.
8	TBTOCINT	W1C	0	GPTM TimerB Time-Out Interrupt Clear 0: The interrupt is unaffected. 1: The interrupt is cleared.
7:4	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
3	RTCCINT	W1C	0	GPTM RTC Interrupt Clear 0: The interrupt is unaffected. 1: The interrupt is cleared.
2	C1ECINT	W1C	0	GPTM Capture1 Event Interrupt Clear 0: The interrupt is unaffected. 1: The interrupt is cleared.
1	C1MCINT	W1C	0	GPTM Capture1 Match Raw Interrupt This is the Capture1 Match interrupt status after masking.
0	TATOCINT	W1C	0	GPTM TimerA Time-Out Raw Interrupt 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 **TimerB Interval Load (GPTMTBILR)** register). In 16-bit mode, the upper 16 bits of this register read as 0's and have no effect on the state of **GPTMTBILR**.

GPTM TimerA Interval Load (GPTMTAILR)
Offset 0x028



1/0 = 1 if timer is configured in 32-bit mode; 0 if timer is configured in 16-bit mode.

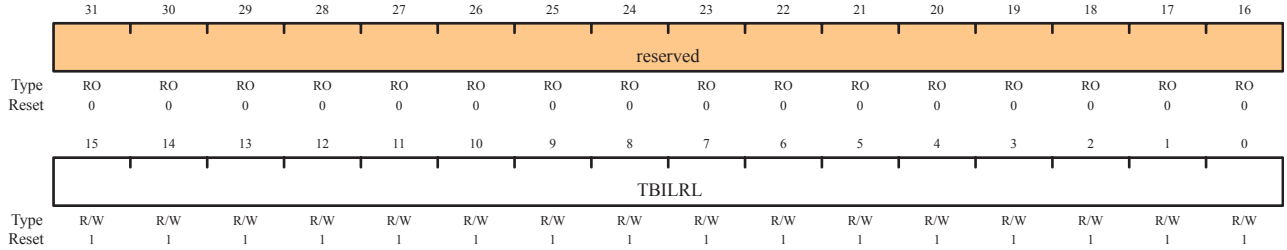
Bit	Name	Type	Reset	Description
31:16	TAILRH	R/W	0xFFFF (32-bit mode) 0x0000 (16-bit mode)	GPTM TimerA Interval Load Register High When configured for 32-bit mode via the **GPTMCFG** register, the **TimerB Interval Load (GPTMTBILR)** register loads this value on a write. A read returns the current value 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)

Offset 0x02C



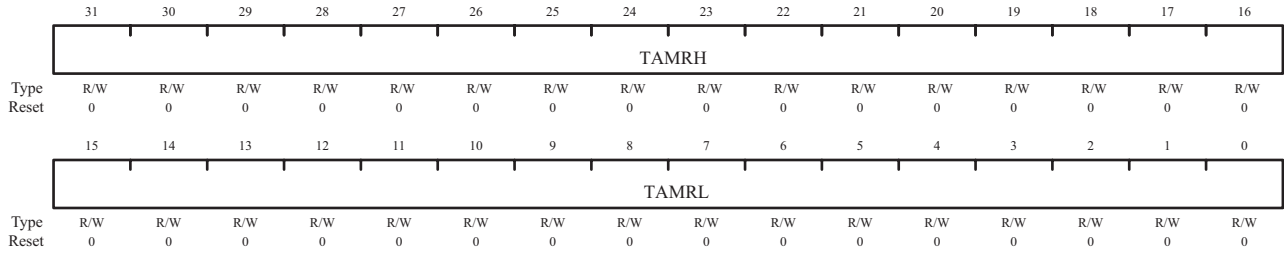
Bit	Name	Type	Reset	Description
31:16	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
15:0	TBILRL	R/W	0xFFFF	GPTM TimerB Interval Load Register When the GPTM is not configured as a 32-bit timer, a write to this field updates **GPTMTBILR**. In 32-bit mode, writes 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)

Offset 0x030



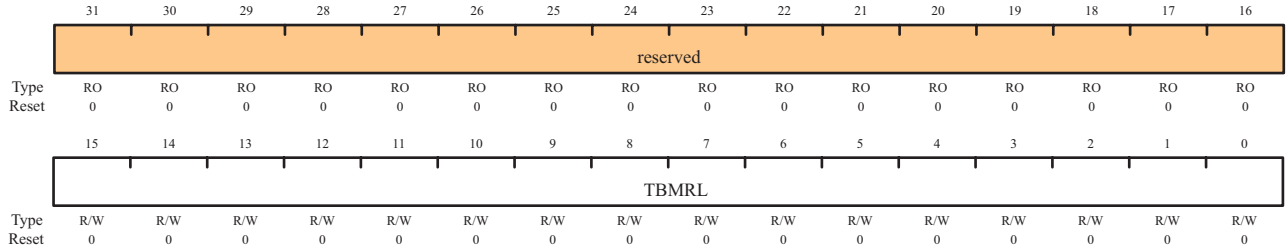
Bit	Name	Type	Reset	Description
31:16	TAMRH	R/W	0xFFFF (32-bit mode) 0x0000 (16-bit mode)	GPTM TimerA Match Register High When configured for 32-bit Real-Time Clock (RTC) mode via the **GPTMCFG** register, this value is compared to the upper half of **GPTMTAR**, to determine match events. In 16-bit mode,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

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

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

GPTM TimerB Match (GPTMTBMATCHR)

Offset 0x034



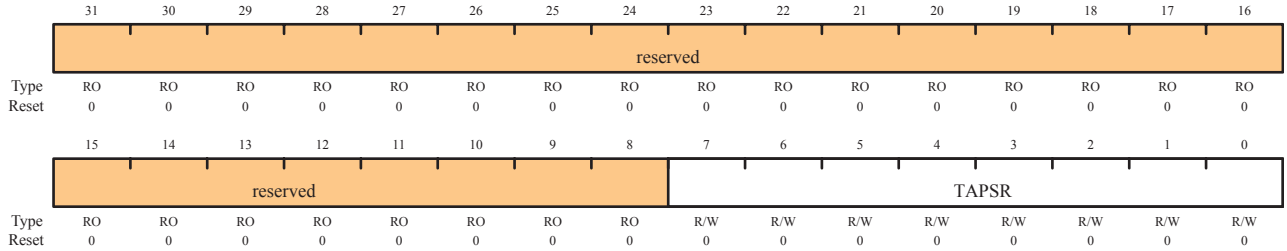
Bit	Name	Type	Reset	Description
31:16	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
15:0	TBMRL	R/W	0xFFFF	GPTM TimerB Match Register Low When configured for PWM mode, this value along with **GPTMTBILR**, determines the duty cycle of the output PWM signal. When 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.

GPTM TimerA Prescale (GPTMTAPR)

Offset 0x038



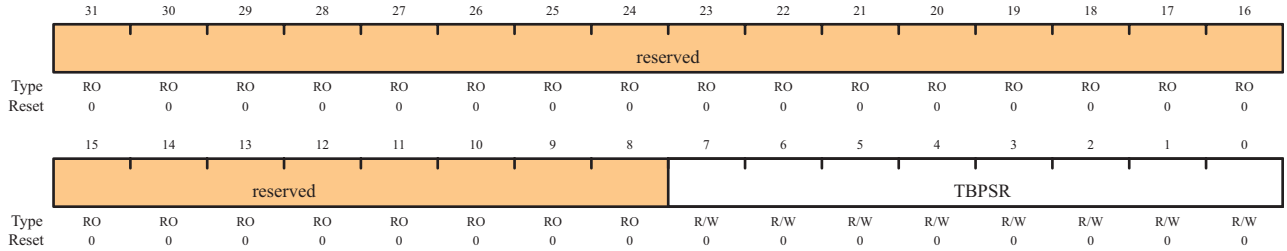
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	TAPSR	R/W	0	GPTM TimerA Prescale The register loads this value on a write. A read returns the current value of the register. Refer to Table 9-1 on page 133 for more details and an example.

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

This register allows software to extend the range of the 16-bit timers.

GPTM TimerB Prescale (GPTMTBPR)

Offset 0x03C



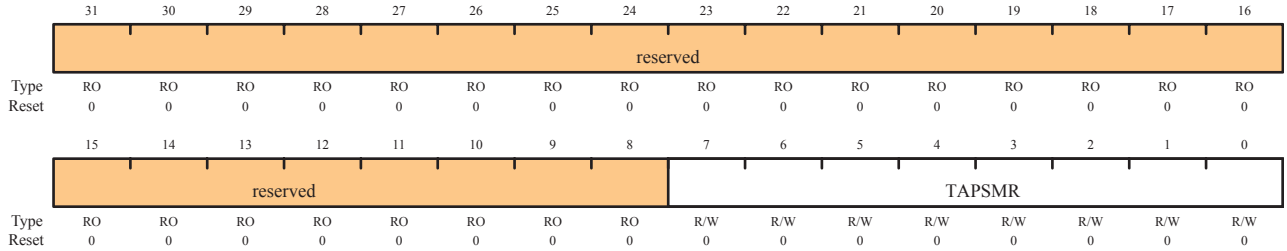
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	TBPSR	R/W	0	GPTM TimerB Prescale The register loads this value on a write. A read returns the current value of this register. Refer to Table 9-1 on page 133 for more details and an example.

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

This register effectively extends the range of **GPTMTAMATCHR** to 24 bits.

GPTM TimerA Prescale Match (GPTMTAPMR)

Offset 0x040



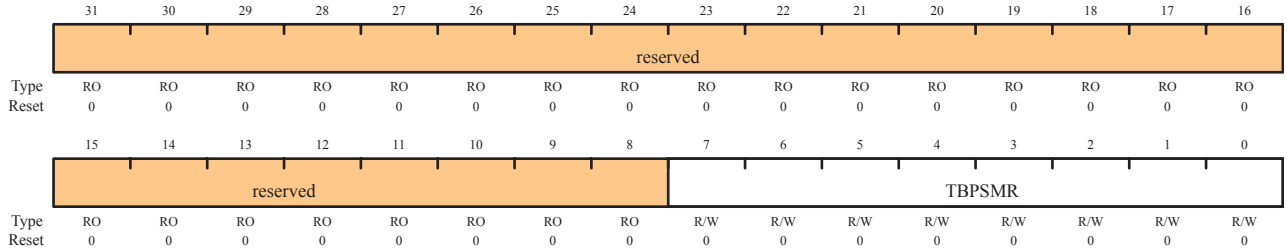
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	TAPSMR	R/W	0	GPTM TimerA Prescale Match This value is used alongside **GPTMTAMATCHR** to detect timer match events while using a prescaler.

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

This register effectively extends the range of **GPTMTBMATCHR** to 24 bits.

GPTM TimerB Prescale Match (GPTMTBPMR)

Offset 0x044



Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	TBPSMR	R/W	0	GPTM TimerB Prescale Match This value is used alongside **GPTMTBMATCHR** to detect timer match events while using a prescaler.

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

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

GPTM TimerA (GPTMTAR)

Offset 0x048

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

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

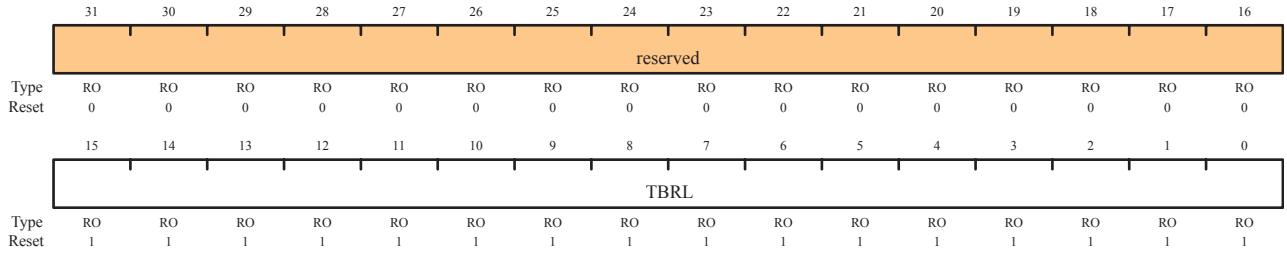
1/0 = 1 if timer is configured in 32-bit mode; 0 if timer is configured in 16-bit mode.

Bit	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 **TimerA Count Register**, except in Input Edge Count mode, when it returns the timestamp from the last edge event.

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

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

GPTM TimerB (GPTMTBR)
Offset 0x04C



Bit	Name	Type	Reset	Description
31:16	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
15:0	TBRL	RO	0xFFFF	GPTM TimerB A read returns the current value of the **TimerB Count Register**, except in Input Edge Count mode, when it returns the timestamp from the last edge event.

10 Watchdog Timer

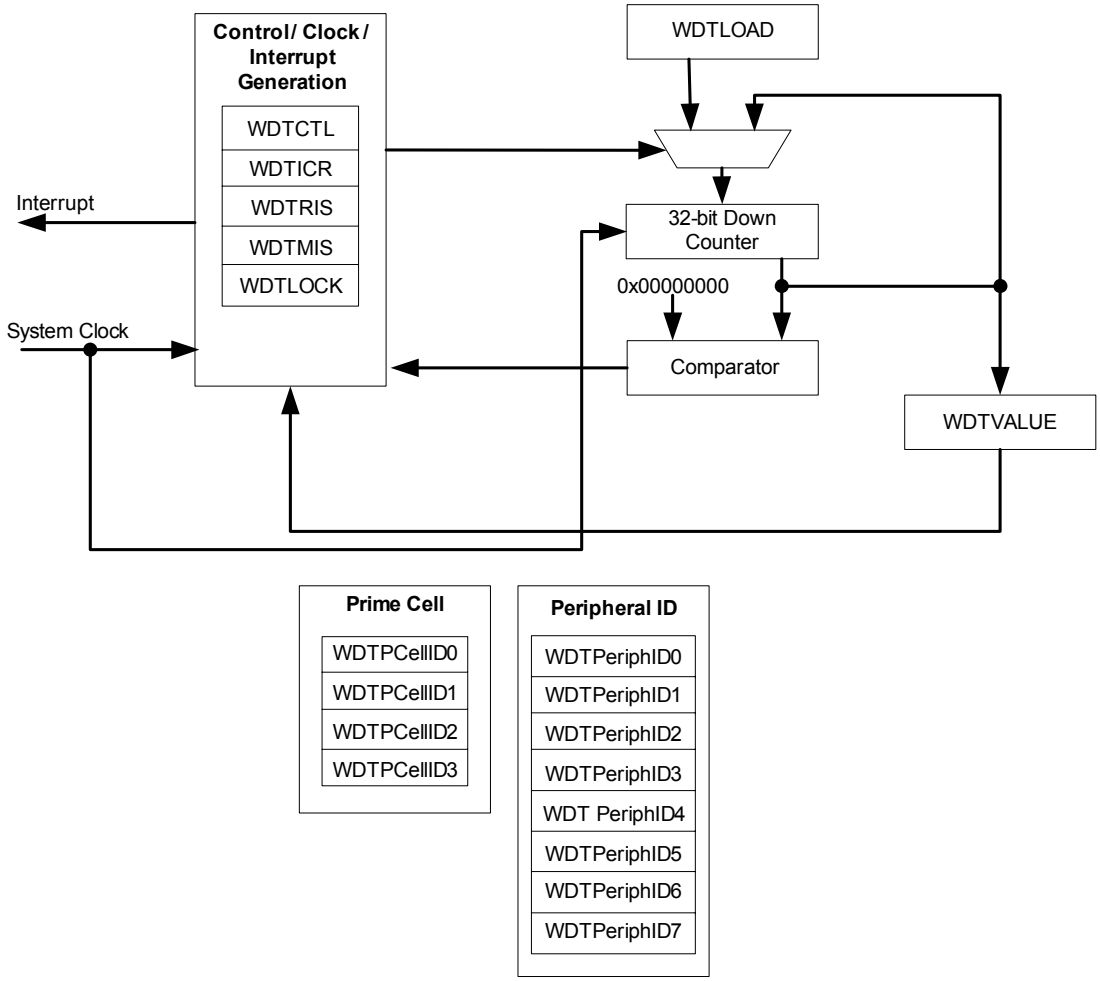
A watchdog timer can generate nonmaskable interrupts (NMIs) or a reset when a time-out value is reached. The watchdog timer is used to regain control when a system has failed due to a software error or to the failure of an external device to respond in the expected way.

The LM3S101 controller 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.

10.1 Block Diagram

Figure 10-1. Watchdog Timer Block Diagram



10.2 Functional Description

The Watchdog Timer module consists of a 32-bit down counter, a programmable load register, interrupt generation logic, and a locking register. Once the Watchdog Timer has been configured, the **Watchdog Timer Lock (WDTLOCK)** register is written, which prevents the timer configuration from being inadvertently altered by software.

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

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

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

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

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

10.3 Initialization and Configuration

The Watchdog Timer is configured using the following sequence:

1. Load the **WDTRLR** register with the desired timer load value.
2. If the Watchdog will be configured to trigger system resets, set the `RESEN` bit in the **WDTCTL** register.
3. Set the `INTEN` bit in the **WDTCTL** register to enable the Watchdog and lock the control register.

If software requires that all of the watchdog registers are locked, the Watchdog Timer module can be fully locked by writing any value to the **WDTLOCK** register. To unlock the Watchdog Timer, write a value of 0x1ACCE551.

10.4 Register Map

Table 10-1 lists the Watchdog registers. All addresses given are relative to the Watchdog Timer base address of 0x40000000.

Table 10-1. WDT Register Map

Offset	Name	Reset	Type	Description	See page
0x000	WDTLOAD	0xFFFFFFFF	R/W	Load	163
0x004	WDTVALUE	0xFFFFFFFF	RO	Current value	164
0x008	WDTCTL	0x00000000	R/W	Control	165
0x00C	WDTICR	-	WO	Interrupt clear	166

Table 10-1. WDT Register Map

Offset	Name	Reset	Type	Description	See page
0x010	WDTRIS	0x00000000	RO	Raw interrupt status	167
0x014	WDTMIS	0x00000000	RO	Masked interrupt status	168
0xC00	WDTLOCK	0x00000000	R/W	Lock	169
0xFD0	WDTPeriphID4	0x00000000	RO	Peripheral identification 4	170
0xFD4	WDTPeriphID5	0x00000000	RO	Peripheral identification 5	171
0xFD8	WDTPeriphID6	0x00000000	RO	Peripheral identification 6	172
0xFDC	WDTPeriphID7	0x00000000	RO	Peripheral identification 7	173
0xFE0	WDTPeriphID0	0x00000005	RO	Peripheral identification 0	174
0xFE4	WDTPeriphID1	0x00000018	RO	Peripheral identification 1	175
0xFE8	WDTPeriphID2	0x00000018	RO	Peripheral identification 2	176
0xFEC	WDTPeriphID3	0x00000001	RO	Peripheral identification 3	177
0xFF0	WDTPCellID0	0x0000000D	RO	PrimeCell identification 0	178
0xFF4	WDTPCellID1	0x000000F0	RO	PrimeCell identification 1	179
0xFF8	WDTPCellID2	0x00000005	RO	PrimeCell identification 2	180
0xFFC	WDTPCellID0	0x000000B1	RO	PrimeCell identification 3	181

10.5 Register Descriptions

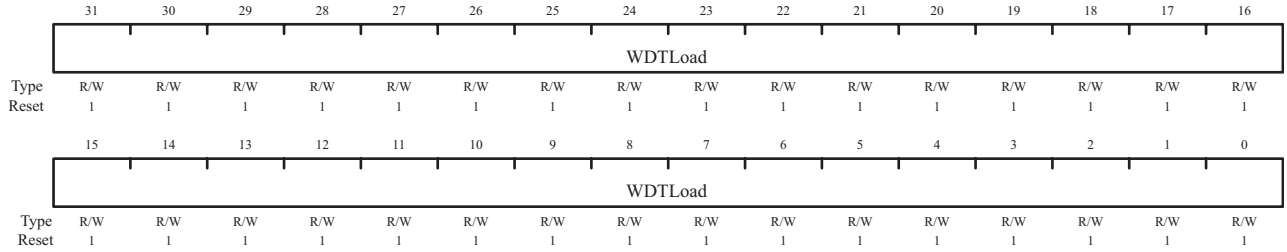
The remainder of this section lists and describes the WDT registers, in numerical order by address offset.

Register 1: Watchdog Load (WDTLOAD), offset 0x000

This register is the 32-bit interval value used by the 32-bit counter. When this register is written, the value is immediately loaded and the counter restarts counting down from the new value. If the **WDTLOAD** register is loaded with 0x00000000, an interrupt is immediately generated.

Watchdog Load (WDTLOAD)

Offset 0x000

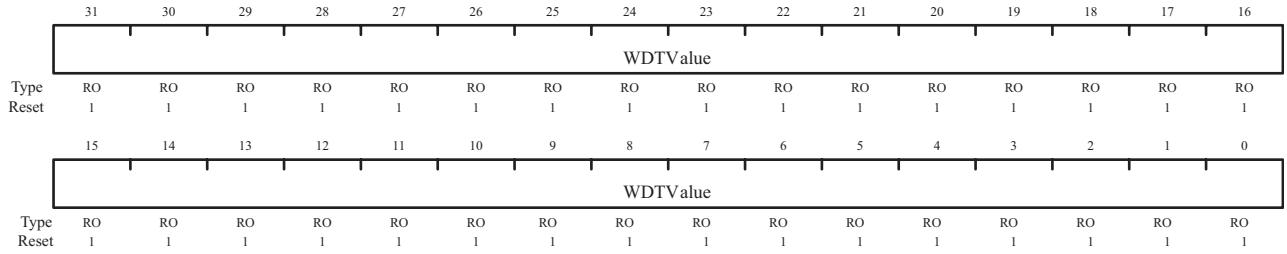


Bit/Field	Name	Type	Reset	Description
31:0	WDTLoad	R/W	0xFFFFFFFF	Watchdog Load Value

Register 2: Watchdog Value (WDTVALUE), offset 0x004

This register contains the current count value of the timer.

Watchdog Value (WDTVALUE)
Offset 0x004



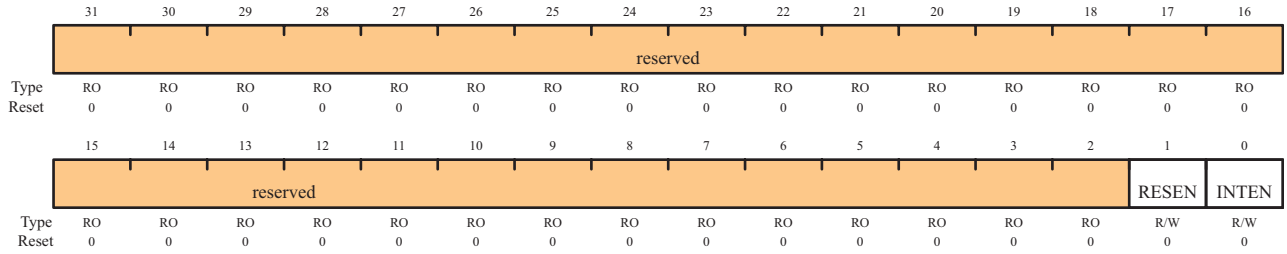
Bit/Field	Name	Type	Reset	Description
31:0	WDTValue	RO	0xFFFFFFFF	Watchdog Value Current value of the 32-bit down counter.

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 (upon 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)
Offset 0x008



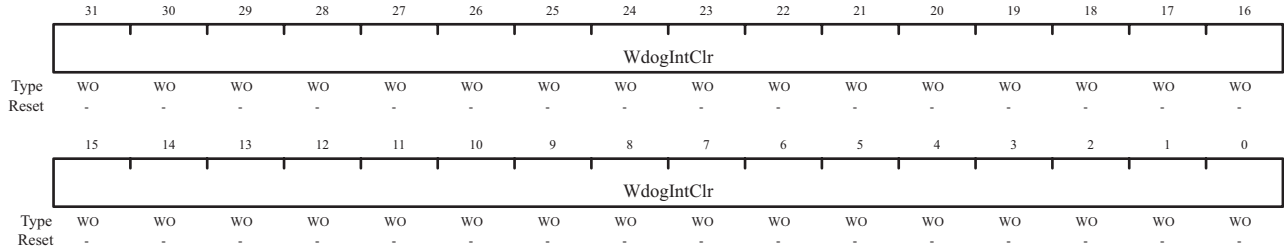
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
1	RESEN	R/W	0x0	Watchdog Reset Enable 0: Disabled. 1: Enable the Watchdog module reset output.
0	INTEN	R/W	0x0	Watchdog Interrupt Enable 0: Interrupt event disabled (once this bit is set, it can only be cleared by a hardware reset) 1: Interrupt event enabled. Once enabled, all writes are ignored.

Register 4: Watchdog Interrupt Clear (WDTICR), offset 0x000

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)

Offset 0x000



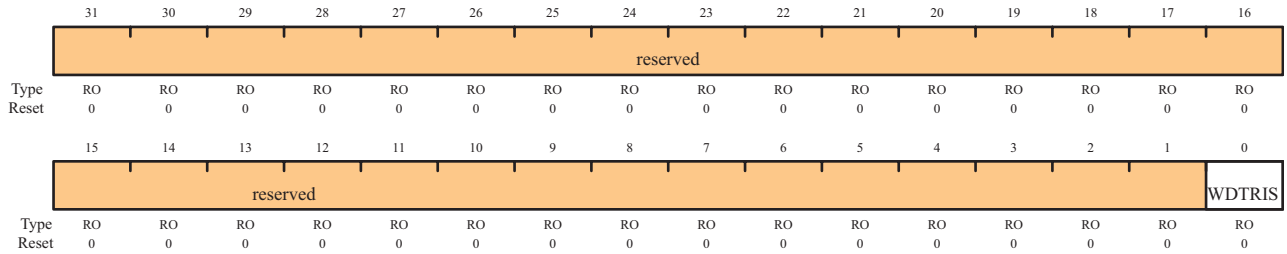
Bit/Field	Name	Type	Reset	Description
31:0	WDTIntClr	WO	-	Watchdog Interrupt Clear

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)

Offset 0x010



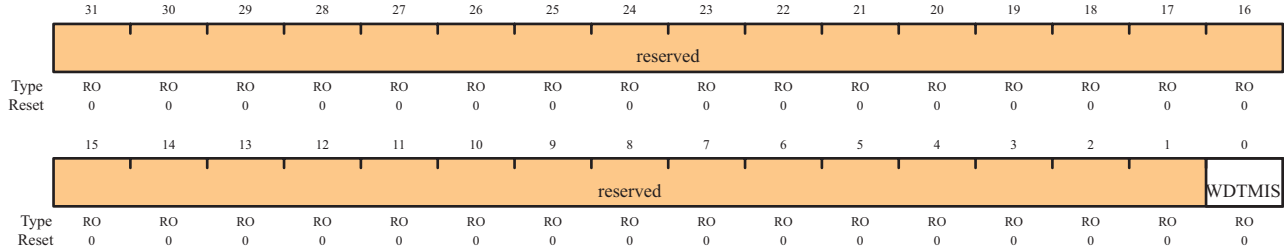
Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
0	WDTRIS	RO	0x0	Watchdog Raw Interrupt Status Gives the raw interrupt state (prior to masking) of **WDTINTR**.

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)

Offset 0x014

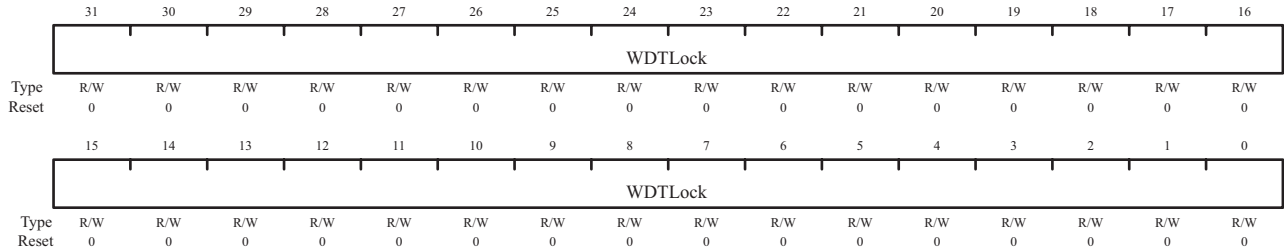


Bit/Field	Name	Type	Reset	Description
31:1	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
0	WDTMIS	RO	0x0	Watchdog Masked Interrupt Status Gives the masked interrupt state (after masking) of the **WDTINTR** interrupt.

Register 7: Watchdog Lock (WDTLOCK), offset 0xC00

Writing 0x1ACCE551 to the **WDTLOCK** register enables write access to all other registers. Writing any other value to the **WDTLOCK** register re-enables the locked state for register writes to all the other registers. Reading the **WDTLOCK** register returns the lock status rather than the 32-bit value written. Therefore, when write accesses are disabled, reading the **WDTLOCK** register returns 0x00000001 (when locked; otherwise, the returned value is 0x00000000 (unlocked)).

Watchdog Lock (WDTLOCK)
Offset 0xC00



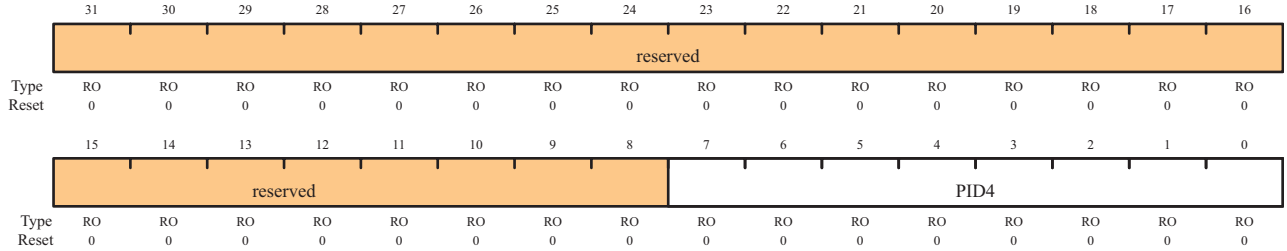
Bit/Field	Name	Type	Reset	Description
31:0	WDTLock	R/W	0x0000	Watchdog Lock A write of the value 0x1ACCE551 unlocks the watchdog registers for write access. A write of any other value reapplies the lock, preventing any register updates. A read of this register returns the following values: Locked: 0x00000001 Unlocked: 0x00000000

Register 8: 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)

Offset 0xFD0

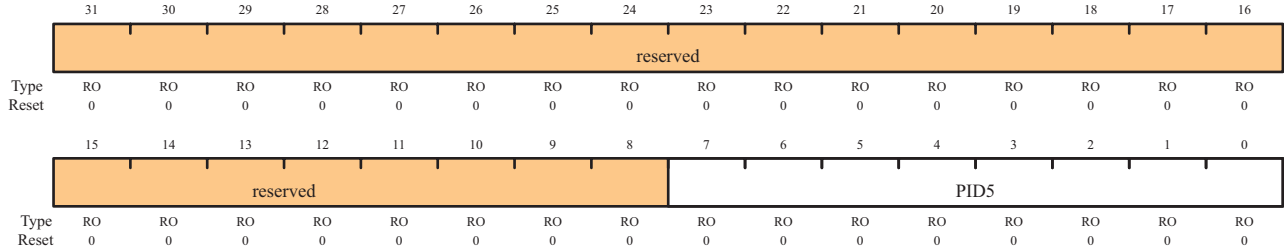


Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID4	RO	0x00	WDT Peripheral ID Register[7:0]

Register 9: 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)
Offset 0xFD4



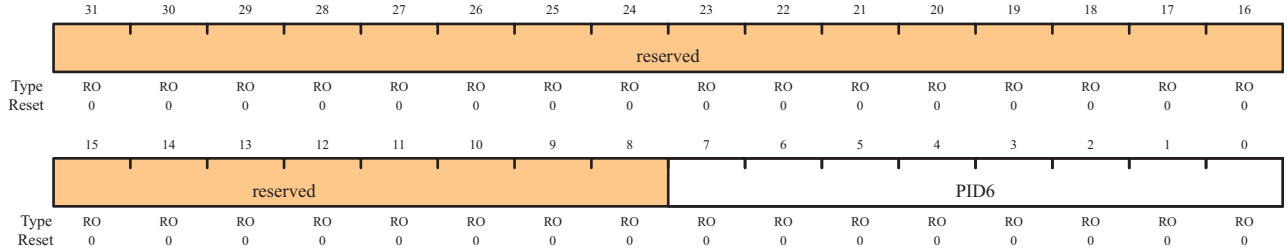
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID5	RO	0x00	WDT Peripheral ID Register[15:8]

Register 10: 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)

Offset 0xFD8



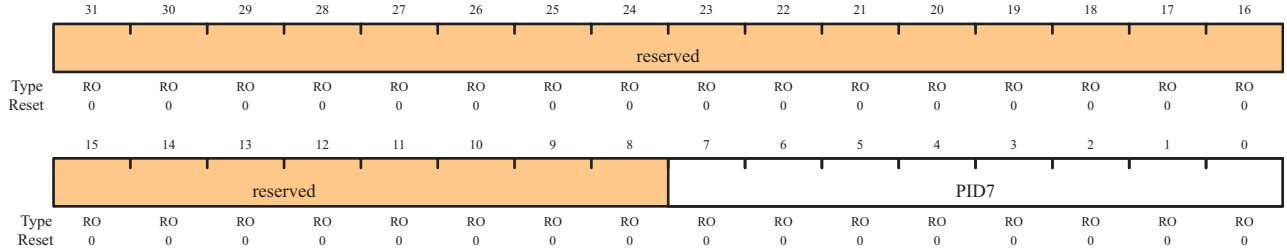
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID6	RO	0x00	WDT Peripheral ID Register[23:16]

Register 11: 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)

Offset 0xFDC



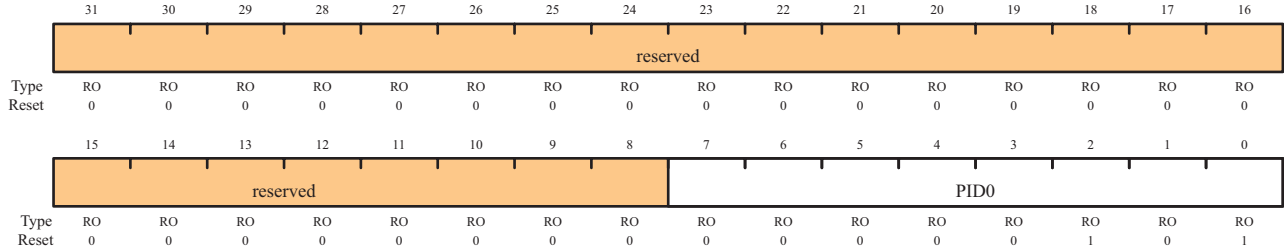
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID7	RO	0x00	WDT Peripheral ID Register[31:24]

Register 12: 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)

Offset 0xFE0

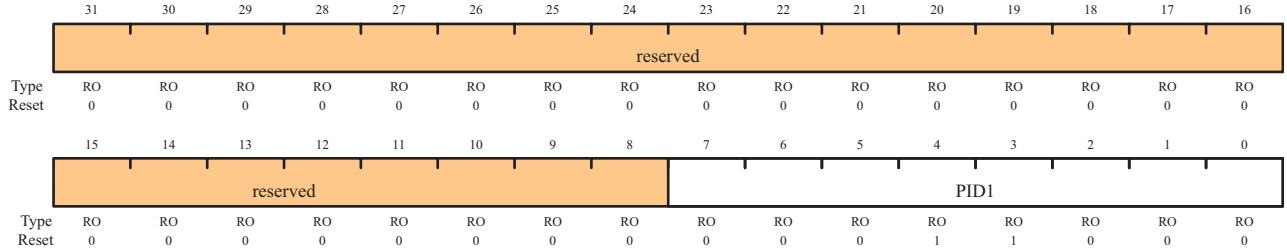


Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID0	RO	0x05	Watchdog Peripheral ID Register[7:0]

Register 13: 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)
Offset 0xFE4



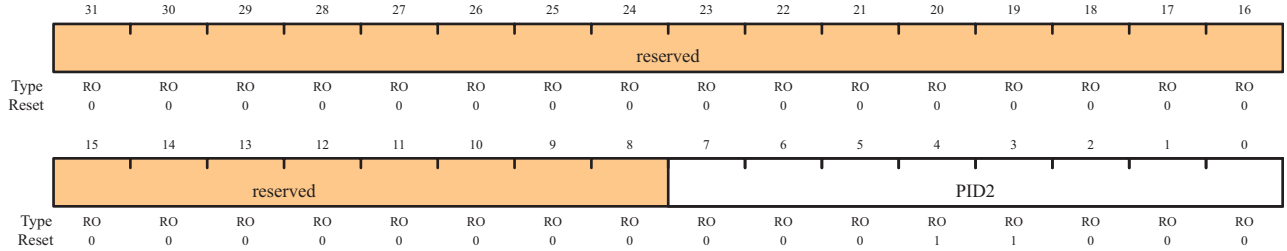
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID1	RO	0x18	Watchdog Peripheral ID Register[15:8]

Register 14: 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)

Offset 0xFE8



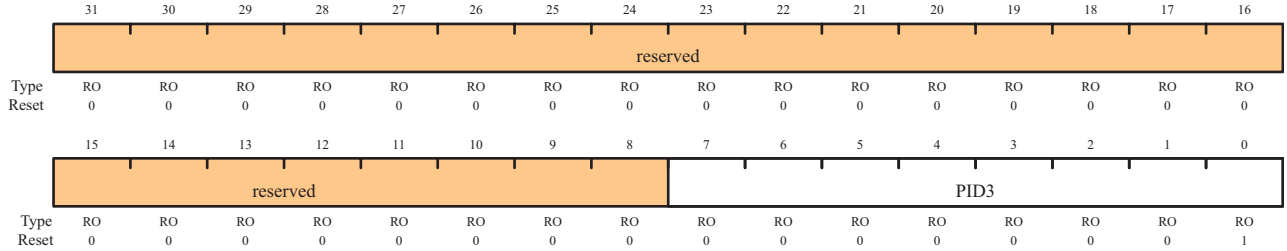
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID2	RO	0x18	Watchdog Peripheral ID Register[23:16]

Register 15: 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)

Offset 0xFEC



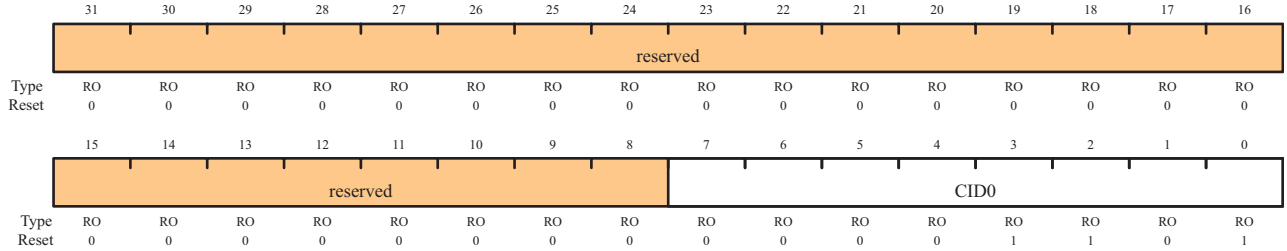
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID3	RO	0x01	Watchdog Peripheral ID Register[31:24]

Register 16: 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)

Offset 0xFF0



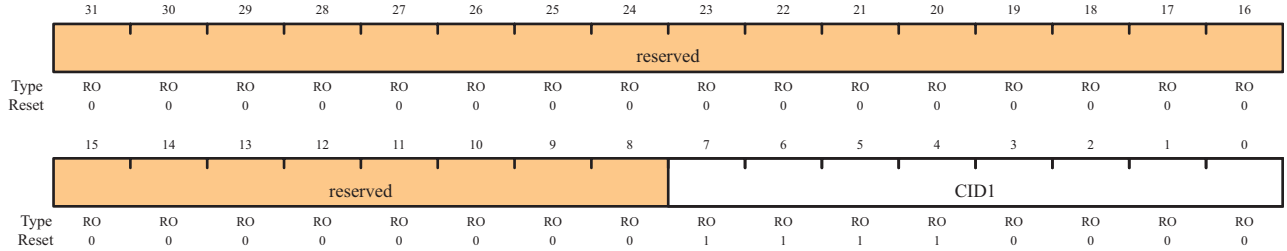
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CID0	RO	0x0D	Watchdog PrimeCell ID Register[7:0]

Register 17: 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)

Offset 0xFF4



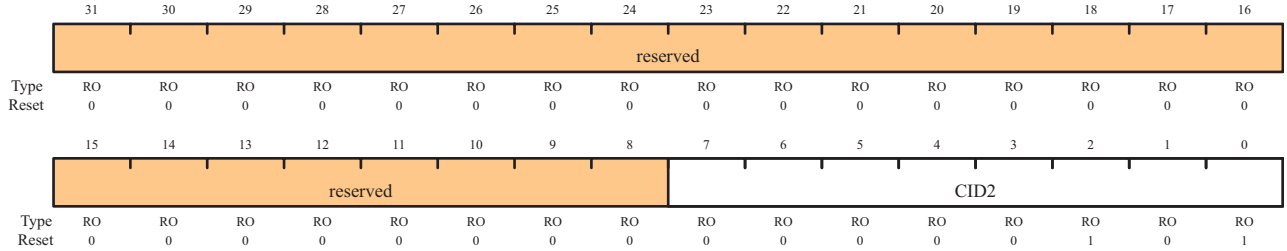
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CID1	RO	0xF0	Watchdog PrimeCell ID Register[15:8]

Register 18: 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)

Offset 0xFF8



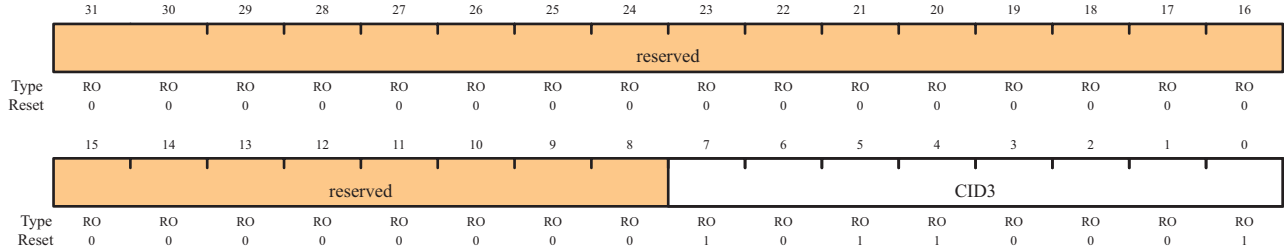
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CID2	RO	0x05	Watchdog PrimeCell ID Register[23:16]

Register 19: Watchdog PrimeCell Identification 3 (WDTPCellID0), offset 0xFFC

The **WDTPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

Watchdog Primecell Identification 3 (WDTPCellID3)

Offset 0xFFC



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CID3	RO	0xB1	Watchdog PrimeCell ID Register[15:8]

11 Universal Asynchronous Receiver/Transmitter (UART)

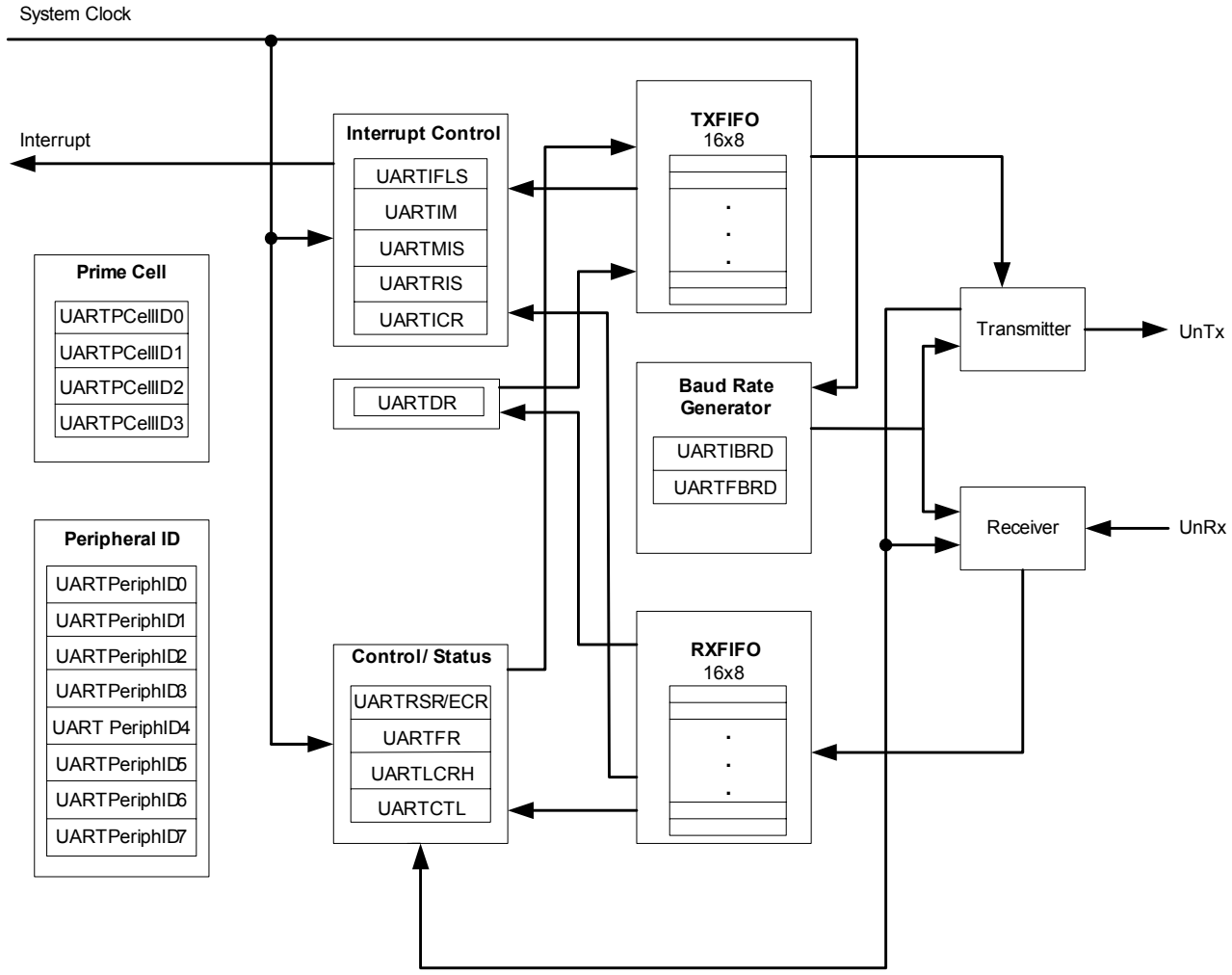
The Universal Asynchronous Receiver/Transmitter (UART) provides fully programmable, 16C550-type serial interface characteristics. The LM3S101 controller is equipped with one UART module.

The UART has the following features:

- Separate transmit and receive FIFOs
- Programmable FIFO length, including 1-byte deep operation providing conventional double-buffered interface
- FIFO trigger levels of 1/8, 1/4, 1/2, 3/4, and 7/8
- Programmable baud-rate generator allowing rates up to 460.8 Kbps
- Standard asynchronous communication bits for start, stop and parity
- False start bit detection
- Line-break generation and detection
- Fully programmable serial interface characteristics:
  - 5, 6, 7, or 8 data bits
  - Even, odd, stick, or no-parity bit generation/detection
  - 1 or 2 stop bit generation

11.1 Block Diagram

Figure 11-1. UART Block Diagram



11.2 Functional Description

The 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.

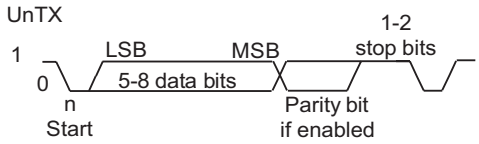
The UART is configured for transmit and/or receive via the TXE and RXE bits of the **UART Control (UARTCTL)** register (see page 199). Transmit and receive are both enabled out of reset. Before any control registers are programmed, the UART must be disabled by clearing the UARTEN bit in **UARTCTL**. If the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

11.2.1 Transmit/Receive Logic

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. The control logic outputs the serial bit stream beginning with a start bit, and followed by the data bits (LSB first), parity bit, and the stop bits according to the programmed configuration in the control registers. See Figure 11-2 for details.

The receive logic performs serial-to-parallel conversion on the received bit stream after a valid start pulse has been detected. Overrun, parity, frame error checking, and line-break detection are also performed, and their status accompanies the data that is written to the receive FIFO.

Figure 11-2. UART Character Frame



11.2.2 Baud-Rate Generation

The baud-rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. The number formed by these two values is used by the baud-rate generator to determine the bit period. Having a fractional baud-rate divider allows the UART to generate all the standard baud rates.

The 16-bit integer is loaded through the **UART Integer Baud-Rate Divisor (UARTIBRD)** register (see page 195) and the 6-bit fractional part is loaded with the **UART Fractional Baud-Rate Divisor (UARTFBRD)** register (see page 196). The baud-rate divisor has the following relationship to the system clock:

```
BRD (Baud-Rate Divisor) = BRDI + BRDF = SysClk / (16 * Baud Rate)
```

Where *BRDI* is the integer part of the BRD and *BRDF* is the fractional part, separated by a decimal place.

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 197), the **UARTIBRD** and **UARTFBRD** registers form an internal 30-bit register. This internal register is only updated when a write operation to **UARTLCRH** is performed, so any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register for the changes to take effect.

To update the baud-rate registers, there are four possible sequences:

- **UARTIBRD** write, **UARTFBRD** write, and **UARTLCRH** write
- **UARTFBRD** write, **UARTIBRD** write, and **UARTLCRH** write
- **UARTIBRD** write and **UARTLCRH** write
- **UARTFBRD** write and **UARTLCRH** write

11.2.3 Data Transmission

Data received or transmitted is stored in two 16-byte FIFOs, though the receive FIFO has an extra four bits per character for status information. For transmission, data is written into the transmit FIFO. If the UART is enabled, it causes a data frame to start transmitting with the parameters

indicated in the **UARTLCRH** register. Data continues to be transmitted until there is no data left in the transmit FIFO. The BUSY bit in the **UART Flag (UARTFR)** register (see page 193) 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 U0Rx 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 183).

The start bit is valid if U0Rx 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 191). 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 U0Rx is High, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO, with any error bits associated with that word.

11.2.4 FIFO Operation

The UART has two 16-entry FIFOs; one for transmit and one for receive. Both FIFOs are accessed via the **UART Data (UARTDR)** register (see page 189). 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 197).

FIFO status can be monitored via the **UART Flag (UARTFR)** register (see page 193) 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 200). Both FIFOs can be individually configured to trigger interrupts at different levels. Available configurations include 1/8, 1/4, 1/2, 3/4 and 7/8. Foe example, if the 1/4 option is selected for the receive FIFO, the UART generates a receive interrupt after 4 data bytes are received. Out of reset, both FIFOs are configured to trigger an interrupt at the 1/2 mark.

11.2.5 Interrupts

The UART can generate interrupts when the following conditions are observed:

- Overrun Error
- Break Error
- Parity Error
- Framing Error
- Receive Timeout
- Transmit (when condition defined in the TXIFLSEL bit in the **UARTIFLS** register is met)
- Receive (when condition defined in the RXIFLSEL bit in the **UARTIFLS** register is met)

All of the interrupt events are ORed together before being sent to the interrupt controller, so the UART can only generate a single interrupt request to the controller at any given time. Software can service multiple interrupt events in a single interrupt service routine by reading the **UART Masked Interrupt Status (UARTMIS)** register (see page 204).

The interrupt events that can trigger a controller-level interrupt are defined in the **UART Interrupt Mask (UARTIM)** register (see page 201) 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 203).

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 205).

11.2.6 Loopback Operation

The UART can be placed into an internal loopback mode for diagnostic or debug work. This is accomplished by setting the LBE bit in the **UARTCTL** register (see page 199). In loopback mode, data transmitted on U0Tx is received on the U0Rx input.

11.3 Initialization and Configuration

This section discusses the steps that are required for using a UART module. For this example, the system clock is assumed to be 20 MHz and the desired UART configuration is:

- 115200 baud rate
- Data length of 8 bits
- One stop bit
- No parity
- FIFOs disabled
- No interrupts

The first thing to consider when programming the UART is the baud-rate divisor (BRD), since the **UARTIBRD** and **UARTFBRD** registers must be written before the **UARTLCRH** register. Using the equation described in “Baud-Rate Generation” on page 184, 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 195) should be set to 10. The value to be loaded into the **UARTFBRD** register (see page 196) 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 0x00000060).
5. Enable the UART by setting the UARTEN bit in the **UARTCTL** register.

11.4 Register Map

Table 11-1 lists the UART registers. All addresses given are relative to the UART's base address:

- UART0: 0x4000C000

Note: The UART must be disabled (see the UARTEN bit in the **UARTCTL** register on page 199) before any of the control registers are reprogrammed. When the UART is disabled during a TX or RX operation, the current transaction is completed prior to the UART stopping.

Table 11-1. UART Register Map

Offset	Name	Reset	Type	Description	See page
0x000	UARTDR	0x00000000	R/W	Data	189
0x004	UARTRSR UARTECR	0x00000000	R/W	Receive Status (read) Error Clear (write)	191
0x018	UARTFR	0x00000090	RO	Flag Register (read only)	193
0x024	UARTIBRD	0x00000000	R/W	Integer Baud-Rate Divisor	195
0x028	UARTFBRD	0x00000000	R/W	Fractional Baud-Rate Divisor	196
0x02C	UARTLCRH	0x00000000	R/W	Line Control Register, High byte	197
0x030	UARTCTL	0x00000300	R/W	Control Register	199
0x034	UARTIFLS	0x00000012	R/W	Interrupt FIFO Level Select	200
0x038	UARTIM	0x00000000	R/W	Interrupt Mask	201
0x03C	UARTRIS	0x0000000F	RO	Raw Interrupt Status	203
0x040	UARTMIS	0x00000000	RO	Masked Interrupt Status	204
0x044	UARTICR	0x00000000	W1C	Interrupt Clear	205
0xFD0	UARTPeriphID4	0x00000000	RO	Peripheral identification 4	206
0xFD4	UARTPeriphID5	0x00000000	RO	Peripheral identification 5	207
0xFD8	UARTPeriphID6	0x00000000	RO	Peripheral identification 6	208
0xFDC	UARTPeriphID7	0x00000000	RO	Peripheral identification 7	209
0xFE0	UARTPeriphID0	0x00000011	RO	Peripheral identification 0	210
0xFE4	UARTPeriphID1	0x00000000	RO	Peripheral identification 1	211
0xFE8	UARTPeriphID2	0x00000018	RO	Peripheral identification 2	212
0xFEC	UARTPeriphID3	0x00000001	RO	Peripheral identification 3	213
0xFF0	UARTPCellID0	0x0000000D	RO	PrimeCell identification 0	214
0xFF4	UARTPCellID1	0x000000F0	RO	PrimeCell identification 1	215
0xFF8	UARTPCellID2	0x00000005	RO	PrimeCell identification 2	216
0xFFC	UARTPCellID3	0x000000B1	RO	PrimeCell identification 3	217

11.5 Register Descriptions

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

Register 1: UART Data (UARTDR), offset 0x000

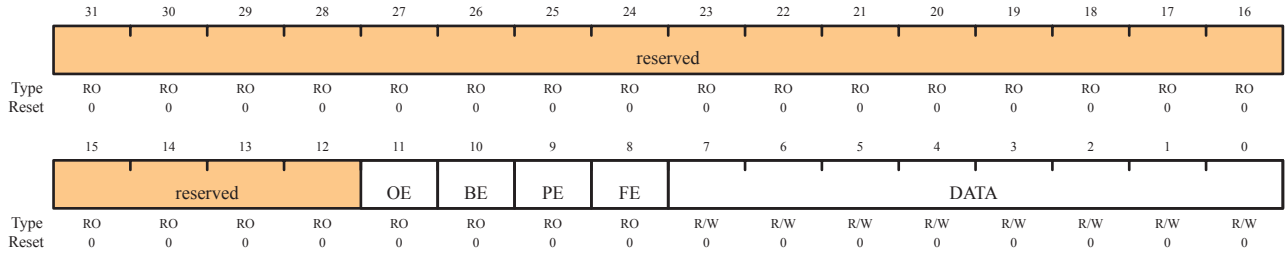
This register is the data register (the interface to the FIFOs).

When FIFOs are enabled, data written to this location is pushed onto the transmit FIFO. If FIFOs are disabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). A write to this register initiates a transmission from the UART.

For received data, if the FIFO is enabled, the data byte and the 4-bit status (break, frame, parity and overrun) is pushed onto the 12-bit wide receive FIFO. If FIFOs are disabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO). The received data can be retrieved by reading this register.

UART Data (UARTDR)

Offset 0x000



Bit	Name	Type	Reset	Description
31:12	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
11	OE	RO	0	UART Overrun Error 1=New data was received when the FIFO was full, resulting in data loss. 0=There has been no data loss due to a FIFO overrun.
10	BE	RO	0	UART Break Error This bit is set to 1 if a break condition was 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 receive data input goes to a 1 (marking state) and the next valid start bit is received.
9	PE	RO	0	UART Parity Error This bit is set to 1 when the parity of the received data character does not match the parity defined by bits 2 and 7 of the **UARTLCRH** register. In FIFO mode, this error is associated with the character at the top of the FIFO.

Bit	Name	Type	Reset	Description
8	FE	RO	0	UART Framing Error When this bit is set to 1, it indicates that the received character did not have a valid stop bit. (A valid stop bit is 1.)
7:0	DATA	R/W	0	When written, the data that is to be transmitted via the UART. When read, the data that was received by the UART.

Register 2: UART Receive Status/Error Clear (UARTRSR/UARTECR), offset 0x004

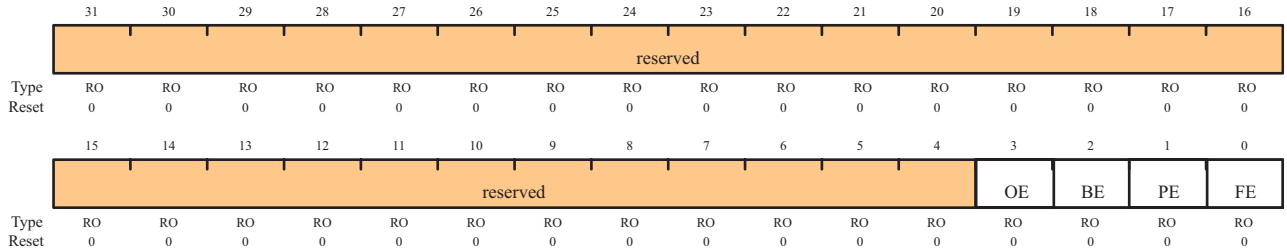
The **UARTRSR/UARTECR** register is the receive status register/error clear register.

In addition to the **UARTDR** register, receive status can also be read from the **UARTRSR** register. If the status is read from this register, then the status information corresponds to the entry read from **UARTDR** prior to reading **UARTRSR**. The status information for overrun is set immediately when an overrun condition occurs.

A write of any value to the **UARTECR** register clears the framing, parity, break, and overrun errors. All the bits are cleared to 0 on reset.

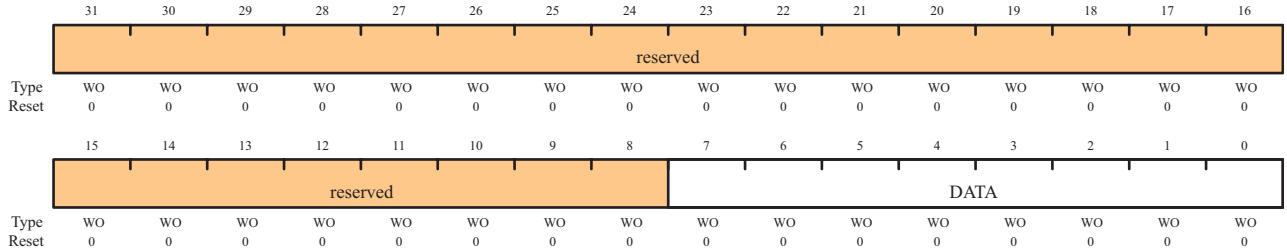
UART Receive Status (UARTRSR): Read

Offset 0x004



UART Error Clear (UARTECR): Write

Offset 0x004



Bit	Name	Type	Reset	Description
Read-Only Receive Status (UARTRSR)				
31:4	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed. The **UARTRSR** register cannot be written.
3	OE	RO	0	UART Overrun Error When this bit is set to 1, data is received and the FIFO is already full. This bit is cleared to 0 by a write to **UARTECR**. The FIFO contents remain valid since no further data is written when the FIFO is full, only the contents of the shift register are overwritten. The CPU must now read the data in order to empty the FIFO. This register cannot be written.

Bit	Name	Type	Reset	Description
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. This register cannot be written.
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**. This register cannot be written.
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. This register cannot be written.

Write-Only Error Clear (UARTECR)

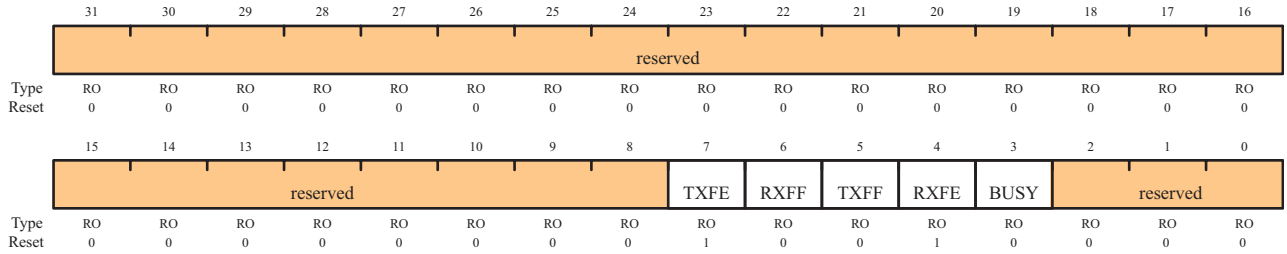
31:8	reserved	WO	0	Reserved bits return an indeterminate value, and should never be changed. The **UARTECR** register cannot be read.
7:0	DATA	WO	0	A write to this register of any data clears the framing, parity, break and overrun flags. The **UARTECR** register cannot be read.

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)

Offset 0x018



Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
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, this bit is set when the transmit holding register is empty. If the FIFO is enabled, 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.

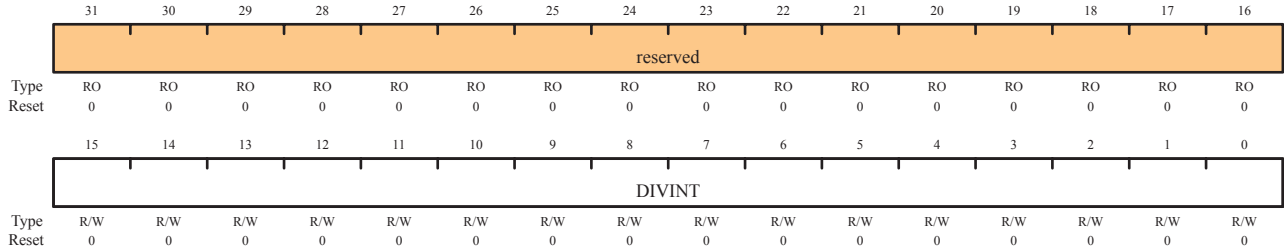
Bit	Name	Type	Reset	Description
4	RXFE	RO	1	UART Receive FIFO Empty The meaning of this bit depends on the state of the FEN bit in the **UARTLCRH** register. If the FIFO is disabled, this bit is set when the receive holding register is empty. If the FIFO is enabled, this bit is set when the receive FIFO 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 or not).
2:0	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.

Register 4: UART Integer Baud-Rate Divisor (UARTIBRD), offset 0x024

The **UARTIBRD** register is the integer part of the baud-rate divisor value. All the bits are cleared on reset. The minimum possible divide ratio is 1 (when **UARTIBRD**=0), in which case the **UARTFBRD** register is ignored. When changing the **UARTIBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 184 for configuration details.

UART Integer Baud-Rate Divisor

Offset 0x024



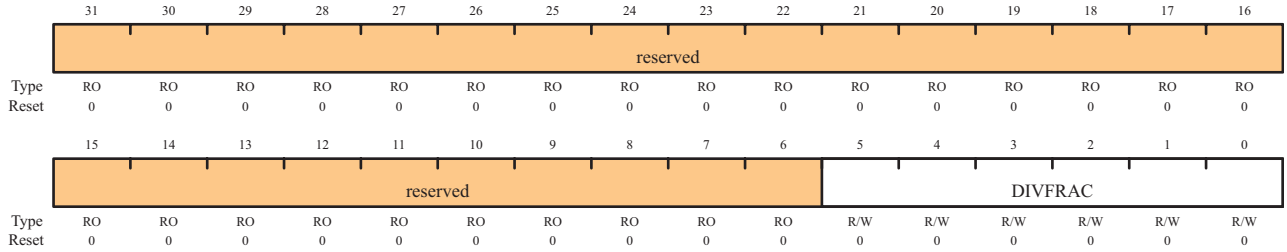
Bit	Name	Type	Reset	Description
31:16	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
15:0	DIVINT	R/W	0x00	Integer Baud-Rate Divisor

Register 5: UART Fractional Baud-Rate Divisor (UARTFBRD), offset 0x028

The **UARTFBRD** register is the fractional part of the baud-rate divisor value. All the bits are cleared on reset. When changing the **UARTFBRD** register, the new value does not take effect until transmission/reception of the current character is complete. Any changes to the baud-rate divisor must be followed by a write to the **UARTLCRH** register. See “Baud-Rate Generation” on page 184 for configuration details.

UART Fractional Baud-Rate Divisor (UARTFBRD)

Offset 0x028



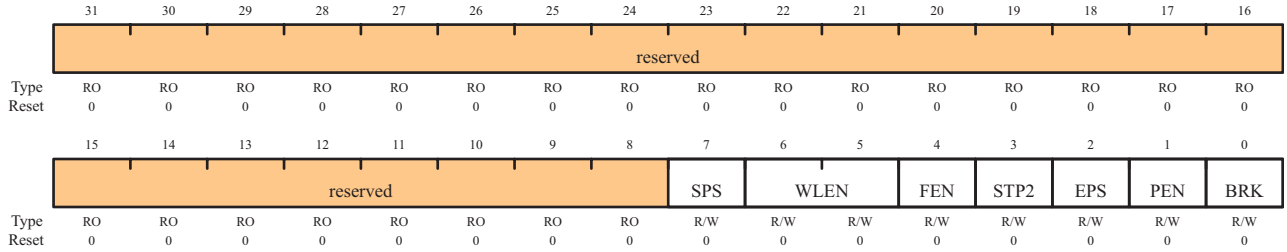
Bit	Name	Type	Reset	Description
31:6	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
5:0	DIVFRAC	R/W	0	Fractional Baud-Rate Divisor

Register 6: UART Line Control (UARTLCRH), offset 0x02C

The **UARTLCRH** register is the line control register. Serial parameters such as data length, parity and stop bit selection are implemented in this register.

When updating the baud-rate divisor (**UARTIBRD** and/or **UARTIFRD**), the **UARTLCRH** register must also be written. The write strobe for the baud-rate divisor registers is tied to the **UARTLCRH** register.

UART Line Control (UARTLCRH)
Offset 0x02C



Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
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: 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.
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.

Bit	Name	Type	Reset	Description
2	EPS	R/W	0	UART Even Parity Select If this bit is set to 1, even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits. When cleared to 0, then odd parity is performed, which checks for an odd number of 1s. This bit has no effect when parity is disabled by the PEN bit.
1	PEN	R/W	0	UART Parity Enable If this bit is set to 1, parity checking and generation is enabled; otherwise, parity is disabled and no parity bit is added to the data frame.
0	BRK	R/W	0	UART Send Break If this bit is set to 1, a Low level is continually output on the UnTX output, after completing transmission of the current character. For the proper execution of the break command, the software must set this bit for at least two frames (character periods). For normal use, this bit must be cleared to 0.

Register 7: UART Control (UARTCTL), offset 0x030

The **UARTCTL** register is the control register. All the bits are cleared on reset except for the `Transmit Enable (TXE)` and `Receive Enable (RXE)` bits, which are set to 1.

To enable the UART module, the `UARTEN` bit must be set to 1. If software requires a configuration change in the module, the `UARTEN` bit must be cleared before the configuration changes are written. If the UART is disabled during a transmit or receive operation, the current transaction is completed prior to the UART stopping.

UART Control (UARTCR)
Offset 0x030

Bit	Name	Type	Reset	Description
31:10	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
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.
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.
7	LBE	R/W	0	UART Loop Back Enable If this bit is set to 1, the `UnTX` path is fed through the `UnRX` path.
6:1	reserved	RO	0	Read as zero.
0	UARTEN	R/W	0	UART Enable If this bit is set to 1, the UART is enabled. When the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.

Register 8: UART Interrupt FIFO Level Select (UARTIFLS), offset 0x034

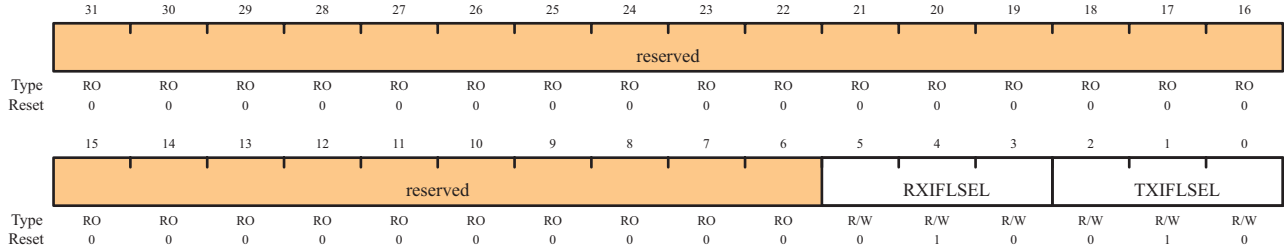
The **UARTIFLS** register is the interrupt FIFO level select register. You can use this register to define the FIFO level at which the TXRIS and RXRIS are triggered.

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)

Offset 0x034



Bit	Name	Type	Reset	Description
31:6	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
5:3	RXIFLSEL	R/W	0X2	UART Receive Interrupt FIFO Level Select 000: RX FIFO ≥ 1/8 full 001: RX FIFO ≥ 1/4 full 010: RX FIFO ≥ 1/2 full (default) 011: RX FIFO ≥ 3/4 full 100: RX FIFO ≥ 7/8 full 101-111: Reserved
2:0	TXIFLSEL	R/W	0X2	UART Transmit Interrupt FIFO Level Select The trigger points for the transmit interrupt are as follows: 000: TX FIFO ≤ 1/8 full 001: TX FIFO ≤ 1/4 full 010: TX FIFO ≤ 1/2 full (default) 011: TX FIFO ≤ 3/4 full 100: TX FIFO ≤ 7/8 full 101-111: Reserved

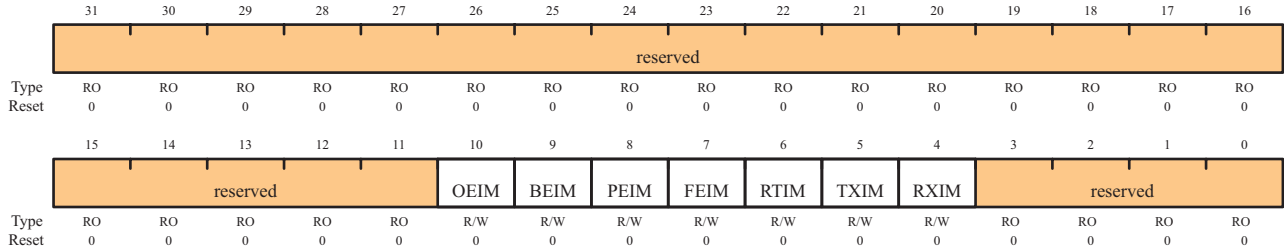
Register 9: UART Interrupt Mask (UARTIM), offset 0x038

The **UARTIM** register is the interrupt mask set/clear register.

On a read, this register gives the current value of the mask on the relevant interrupt. Writing a 1 to a bit allows the corresponding raw interrupt signal to be routed to the interrupt controller. Writing a 0 prevents the raw interrupt signal from being sent to the interrupt controller.

UART Interrupt Mask (UARTIM)

Offset 0x038



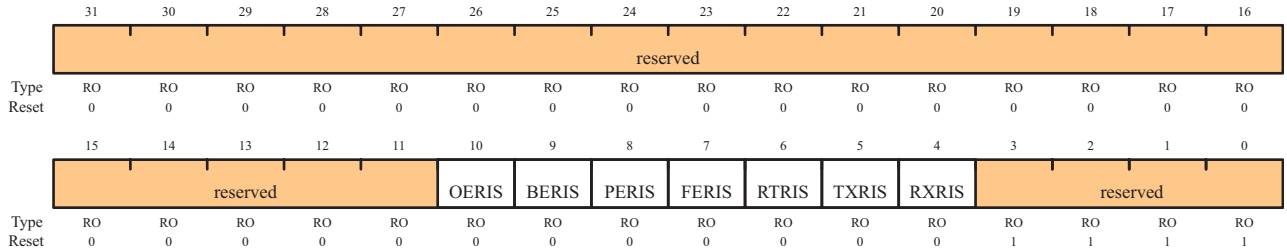
Bit	Name	Type	Reset	Description
31:11	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
10	OEIM	R/W	0	UART Overrun Error Interrupt Mask On a read, the current mask for the `OEIM` interrupt is returned. Setting this bit to 1 promotes the `OEIM` interrupt to the interrupt 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.

Bit	Name	Type	Reset	Description
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.
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	0	Reserved bits return an indeterminate value, and should never be changed.

Register 10: UART Raw Interrupt Status (UARTRIS), offset 0x03C

The **UARTRIS** register is the raw interrupt status register. On a read this register gives the current raw status value of the corresponding interrupt. A write has no effect.

UART Raw Interrupt Status (UARTRIS)
Offset 0x03C



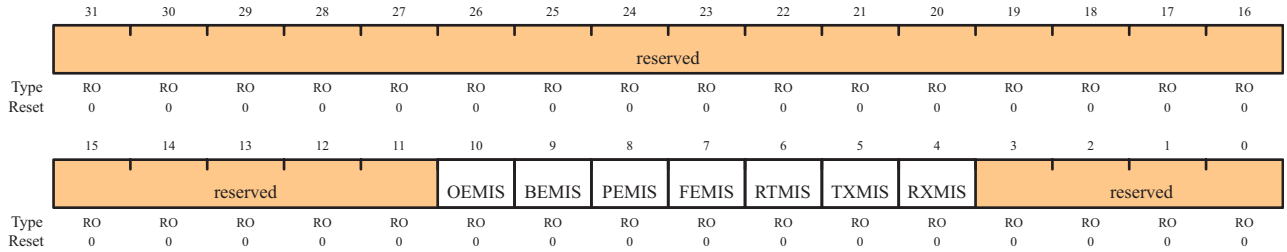
Bit	Name	Type	Reset	Description
31:11	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
10	OERIS	RO	0	UART Overrun Error Raw Interrupt Status Gives the raw interrupt state (prior to masking) of this interrupt.
9	BERIS	RO	0	UART Break Error Raw Interrupt Status Gives 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	This reserved bit is read-only.

Register 11: UART Masked Interrupt Status (UARTMIS), offset 0x040

The **UARTMIS** register is the masked interrupt status register. On a read this register gives the current masked status value of the corresponding interrupt. A write has no effect.

UART Masked Interrupt Status (UARTMIS)

Offset 0x040



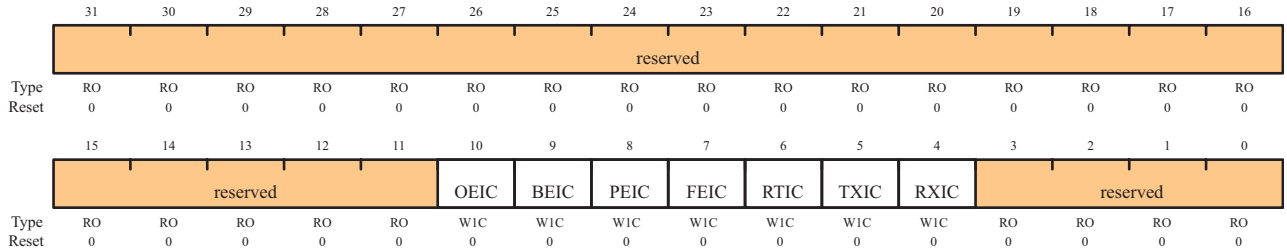
Bit	Name	Type	Reset	Description
31:11	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
10	OEMIS	RO	0	UART Overrun Error Masked Interrupt Status Gives the masked interrupt state of this interrupt.
9	BEMIS	RO	0	UART Break Error Masked Interrupt Status Gives the masked 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	Reserved bits return an indeterminate value, and should never be changed.

Register 12: UART Interrupt Clear (UARTICR), offset 0x044

The **UARTICR** register is the interrupt clear register. On a write of 1, the corresponding interrupt (both raw interrupt and masked interrupt, if enabled) is cleared. A write of 0 has no effect.

UART Interrupt Clear (UARTICR)

Offset 0x044



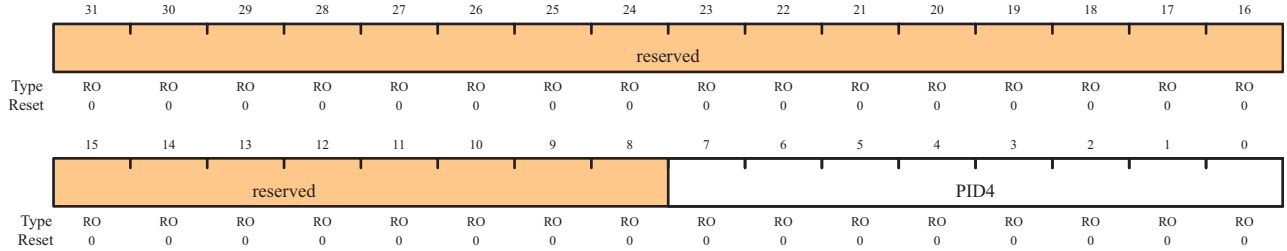
Bit	Name	Type	Reset	Description
31:11	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
10	OEIC	W1C	0	Overrun Error Interrupt Clear 0: No effect on the interrupt 1: Clears interrupt
9	BEIC	W1C	0	Break Error Interrupt Clear 0: No effect on the interrupt 1: Clears interrupt
8	PEIC	W1C	0	Parity Error Interrupt Clear 0: No effect on the interrupt 1: Clears interrupt
7	FEIC	W1C	0	Framing Error Interrupt Clear 0: No effect on the interrupt 1: Clears interrupt
6	RTIC	W1C	0	Receive Time-Out Interrupt Clear 0: No effect on the interrupt 1: Clears interrupt
5	TXIC	W1C	0	Transmit Interrupt Clear 0: No effect on the interrupt 1: Clears interrupt
4	RXIC	W1C	0	Receive Interrupt Clear 0: No effect on the interrupt 1: Clears interrupt
3:0	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.

Register 13: UART Peripheral Identification 4 (UARTPeriphID4), offset 0xFD0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 4 (UARTPeriphID4)

Offset 0xFD0



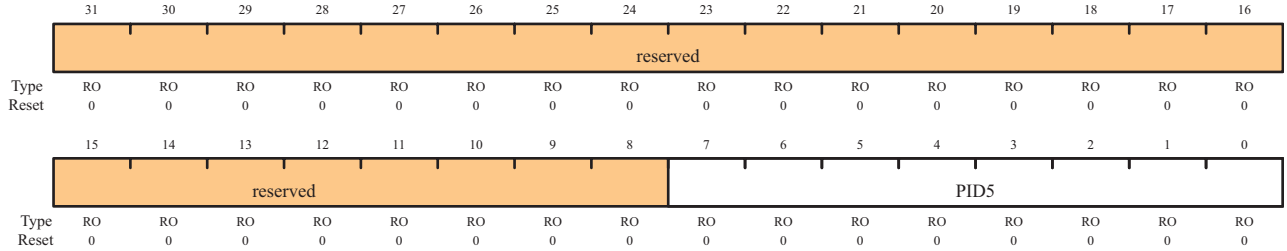
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID4	RO	0x0	UART Peripheral ID Register[7:0]

Register 14: UART Peripheral Identification 5 (UARTPeriphID5), offset 0xFD4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 5 (UARTPeriphID5)

Offset 0xFD4



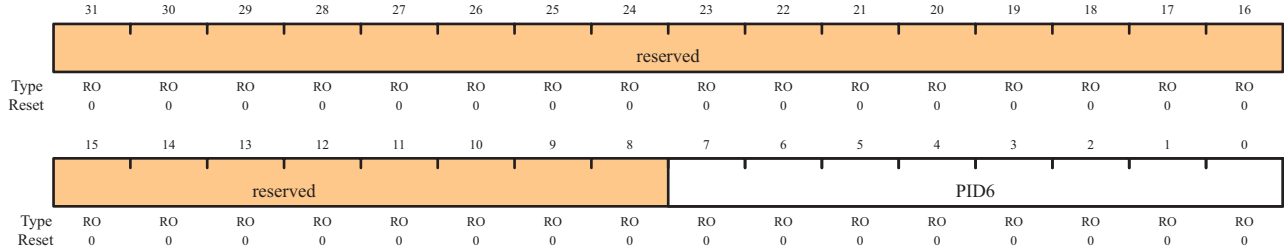
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID5	RO	0x0	UART Peripheral ID Register[15:8]

Register 15: UART Peripheral Identification 6 (UARTPeriphID6), offset 0xFD8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 6 (UARTPeriphID6)

Offset 0xFD8



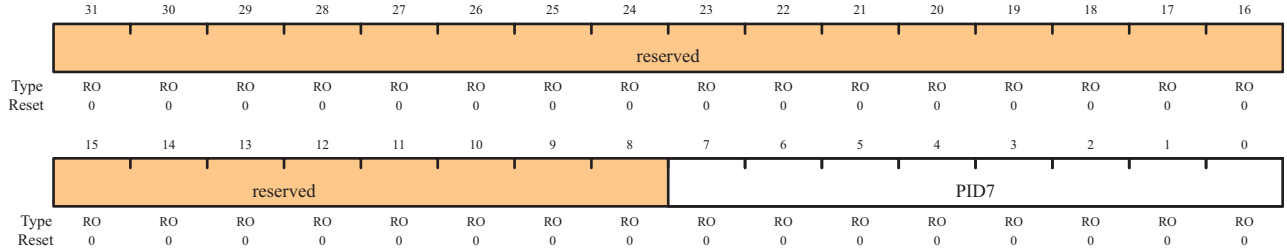
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID6	RO	0x0	UART Peripheral ID Register[23:16]

Register 16: UART Peripheral Identification 7 (UARTPeriphID7), offset 0xFDC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 7 (UARTPeriphID7)

Offset 0xFDC



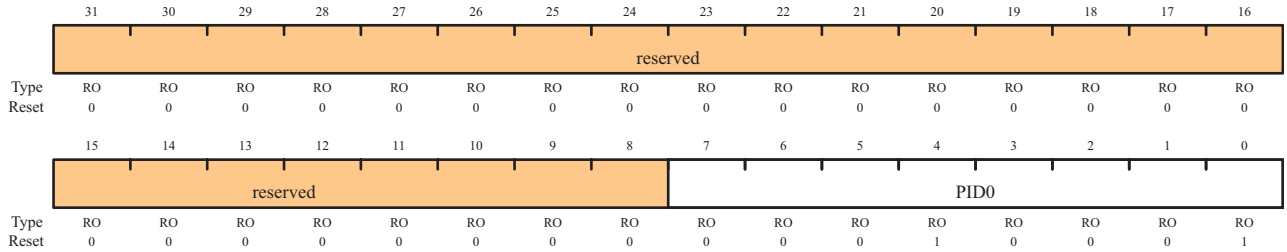
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID7	RO	0x0	UART Peripheral ID Register[31:24]

Register 17: UART Peripheral Identification 0 (UARTPeriphID0), offset 0xFE0

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 0 (UARTPeriphID0)

Offset 0xFE0



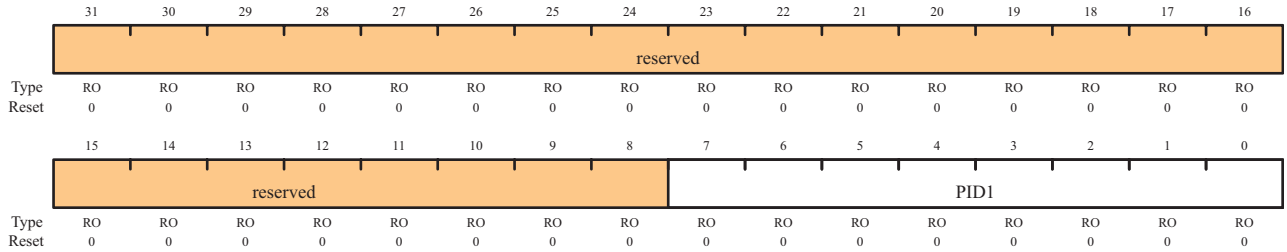
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID0	RO	0x11	UART Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 18: UART Peripheral Identification 1 (UARTPeriphID1), offset 0xFE4

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 1 (UARTPeriphID1)

Offset 0xFE4



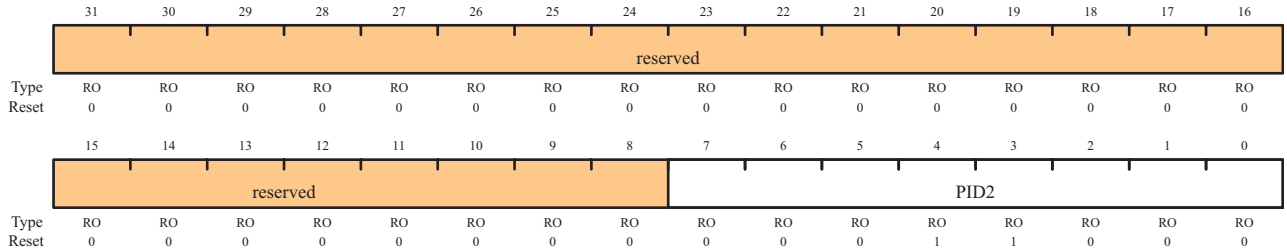
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID1	RO	0x0	UART Peripheral ID Register[15:8] Can be used by software to identify the presence of this peripheral.

Register 19: UART Peripheral Identification 2 (UARTPeriphID2), offset 0xFE8

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 2 (UARTPeriphID2)

Offset 0xFE8



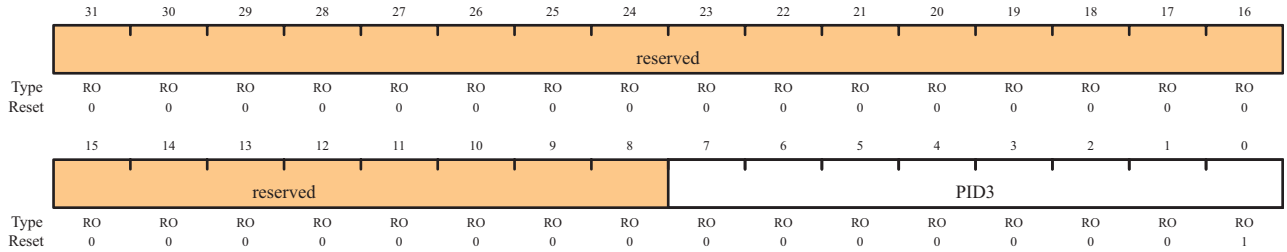
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID2	RO	0x18	UART Peripheral ID Register[23:16] Can be used by software to identify the presence of this peripheral.

Register 20: UART Peripheral Identification 3 (UARTPeriphID3), offset 0xFEC

The **UARTPeriphIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Peripheral Identification 3 (UARTPeriphID3)

Offset 0xFEC



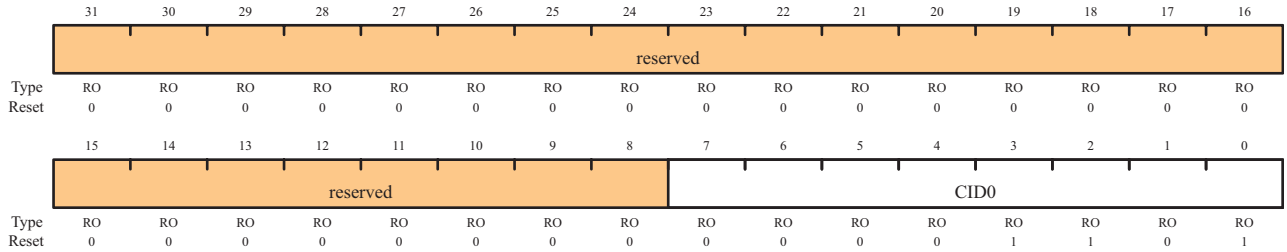
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID3	RO	0x1	UART Peripheral ID Register[31:24] Can be used by software to identify the presence of this peripheral.

Register 21: UART PrimeCell Identification 0 (UARTPCellID0), offset 0xFF0

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Primecell Identification 0 (UARTPCellID0)

Offset 0xFF0



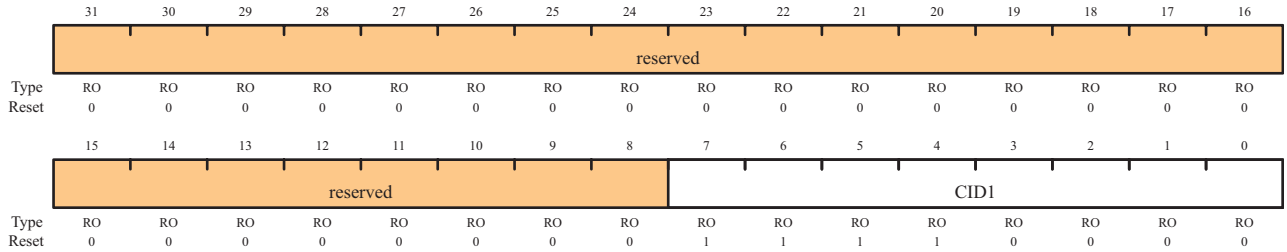
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CID0	RO	0x0D	UART PrimeCell ID Register[7:0] Provides software a standard cross-peripheral identification system.

Register 22: UART PrimeCell Identification 1 (UARTPCellID1), offset 0xFF4

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Primecell Identification 1 (UARTPCellID1)

Offset 0xFF4



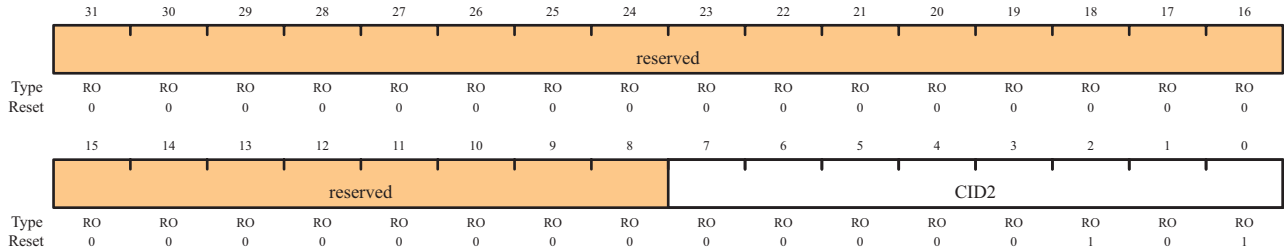
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CID1	RO	0xF0	UART PrimeCell ID Register[15:8] Provides software a standard cross-peripheral identification system.

Register 23: UART PrimeCell Identification 2 (UARTPCellID2), offset 0xFF8

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Primecell Identification 2 (UARTPCellID2)

Offset 0xFF8



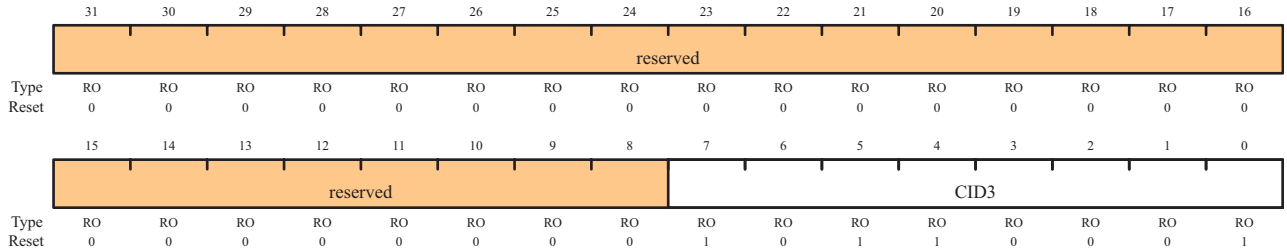
Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CID2	RO	0x05	UART PrimeCell ID Register[23:16] Provides software a standard cross-peripheral identification system.

Register 24: UART PrimeCell Identification 3 (UARTPCellID3), offset 0xFFC

The **UARTPCellIDn** registers are hard-coded and the fields within the registers determine the reset values.

UART Primecell Identification 3 (UARTPCellID3)

Offset 0xFFC



Bit	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CID3	RO	0xB1	UART PrimeCell ID Register[31:24] Provides software a standard cross-peripheral identification system.

12 Synchronous Serial Interface (SSI)

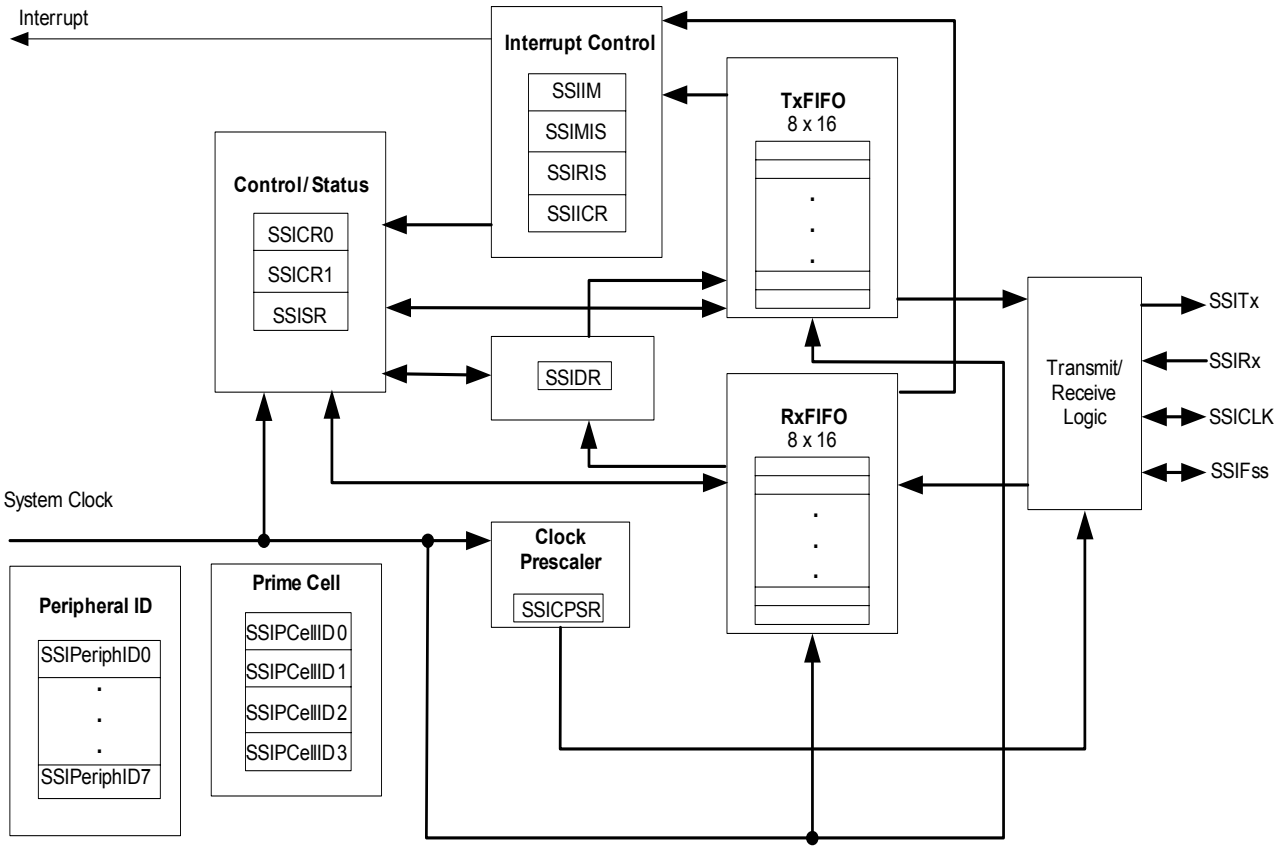
The Synchronous Serial Interface (SSI) is a master or slave interface for synchronous serial communication with peripheral devices that have either Freescale SPI, National Semiconductor MICROWIRE, or Texas Instruments synchronous serial interfaces.

The Stellaris SSI has the following features:

- Master or slave operation
- Programmable clock bit rate and prescale
- Separate transmit and receive FIFOs, 16 bits wide, 8 locations deep
- Programmable interface operation for Freescale SPI, National Semiconductor MICROWIRE, or Texas Instruments synchronous serial interfaces
- Programmable data frame size from 4 to 16 bits
- Internal loopback test mode for diagnostic/debug testing

12.1 Block Diagram

Figure 12-1. SSI Block Diagram



12.2 Functional Description

The SSI performs serial-to-parallel conversion on data received from a peripheral device. The CPU accesses data, control, and status information. The transmit and receive paths are buffered with internal FIFO memories allowing up to eight 16-bit values to be stored independently in both transmit and receive modes.

12.2.1 Bit Rate Generation

The SSI includes a programmable bit rate clock divider and prescaler to generate the serial output clock. Bit rates are supported to 1.5 MHz and higher, although maximum bit rate is determined by peripheral devices.

The serial bit rate is derived by dividing down the 20-MHz input clock. The clock is first divided by an even prescale value `CPSDVSR` from 2 to 254, which is programmed in the **SSI Clock Prescale (SSICPSR)** register (see page 234). 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 229).

The frequency of the output clock `SSIClk` is defined by:

```
FSSIClk = FSysClk / (CPSDVR * (1 + SCR))
```

Note that although the SSIClk transmit clock can theoretically be 10 MHz, the module may not be able to operate at that speed. For transmit operations, the system clock must be at least two times faster than the SSIClk. For receive operations, the system clock must be at least 12 times faster than the SSIClk.

See “Electrical Characteristics” on page 271 to view SSI timing parameters.

12.2.2 FIFO Operation

12.2.2.1 Transmit FIFO

The common transmit FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. The CPU writes data to the FIFO by writing the **SSI Data (SSIDR)** register (see page 232), and data is stored in the FIFO until it is read out by the transmission logic.

When configured as a master or a slave, parallel data is written into the transmit FIFO prior to serial conversion and transmission to the attached slave or master, respectively, through the `SSITx` pin.

12.2.2.2 Receive FIFO

The common receive FIFO is a 16-bit wide, 8-locations deep, first-in, first-out memory buffer. Received data from the serial interface is stored in the buffer until read out by the CPU, which accesses the read FIFO by reading the **SSIDR** register.

When configured as a master or slave, serial data received through the `SSIRx` pin is registered prior to parallel loading into the attached slave or master receive FIFO, respectively.

12.2.3 Interrupts

The SSI can generate interrupts when the following conditions are observed:

- Transmit FIFO service
- Receive FIFO service
- Receive FIFO time-out
- Receive FIFO overrun

All of the interrupt events are ORed together before being sent to the interrupt controller, so the SSI can only generate a single interrupt request to the controller at any given time. You can mask each of the four individual maskable interrupts by setting the appropriate bits in the **SSI Interrupt Mask (SSIIM)** register (see page 235). 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 236 and page 237, respectively).

12.2.4 Frame Formats

Each data frame is between 4 and 16 bits long, depending on the size of data programmed, and is transmitted starting with the MSB. There are three basic frame types that can be selected:

- Texas Instruments synchronous serial
- Freescale SPI
- National Semiconductor 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 National Semiconductor MICROWIRE frame formats, the serial frame (`SSIFss`) pin is active Low, and is asserted (pulled down) during the entire transmission of the frame.

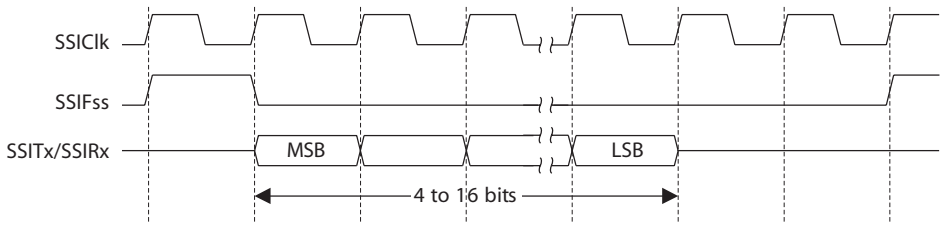
For Texas Instruments synchronous serial frame format, the `SSIFss` pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSI and the off-chip slave device drive their output data on the rising edge of `SSIClk`, and latch data from the other device on the falling edge.

Unlike the full-duplex transmission of the other two frame formats, the National Semiconductor MICROWIRE format uses a special master-slave messaging technique, which operates at half-duplex. In this mode, when a frame begins, an 8-bit control message is transmitted to the off-chip slave. During this transmit, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the requested data. The returned data can be 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

12.2.4.1 Texas Instruments Synchronous Serial Frame Format

Figure 12-2 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

Figure 12-2. TI Synchronous Serial Frame Format (Single Transfer)

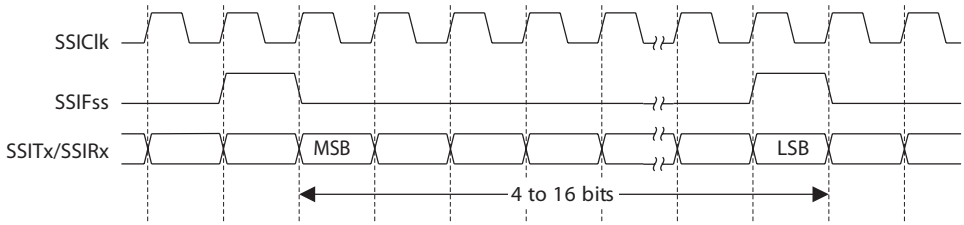


In this mode, SSIClk and SSIFss are forced Low, and the transmit data line SSITx is tristated whenever the SSI is idle. Once the bottom entry of the transmit FIFO contains data, SSIFss is pulsed High for one SSIClk period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of SSIClk, the MSB of the 4 to 16-bit data frame is shifted out on the SSITx pin. Likewise, the MSB of the received data is shifted onto the SSIRx pin by the off-chip serial slave device.

Both the SSI and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each SSIClk. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of SSIClk after the LSB has been latched.

Figure 12-3 shows the Texas Instruments synchronous serial frame format when back-to-back frames are transmitted.

Figure 12-3. TI Synchronous Serial Frame Format (Continuous Transfer)



12.2.4.2 Freescale SPI Frame Format

The Freescale SPI interface is a four-wire interface where the SSIFss signal behaves as a slave select. The main feature of the Freescale SPI format is that the inactive state and phase of the SSIClk signal are programmable through the SPO and SPH bits within the **SSISCR0** control register.

SPO Clock Polarity Bit

When the SPO clock polarity control bit is Low, it produces a steady state Low value on the SSIClk pin. If the SPO bit is High, a steady state High value is placed on the SSIClk pin when data is not being transferred.

SPH Phase Control Bit

The SPH phase control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the SPH phase control bit is Low, data is captured on the first clock edge transition. If the SPH bit is High, data is captured on the second clock edge transition.

12.2.4.3 Freescale SPI Frame Format with SPO=0 and SPH=0

Single and continuous transmission signal sequences for Freescale SPI format with SPO=0 and SPH=0 are shown in Figure 12-4 and Figure 12-5.

Figure 12-4. Freescale SPI Format (Single Transfer) with SPO=0 and SPH=0

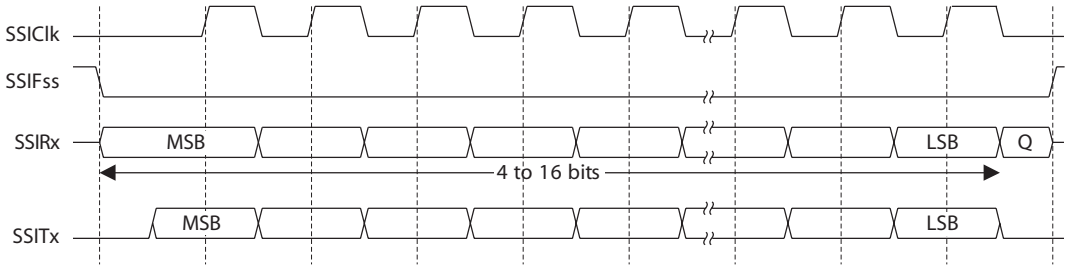
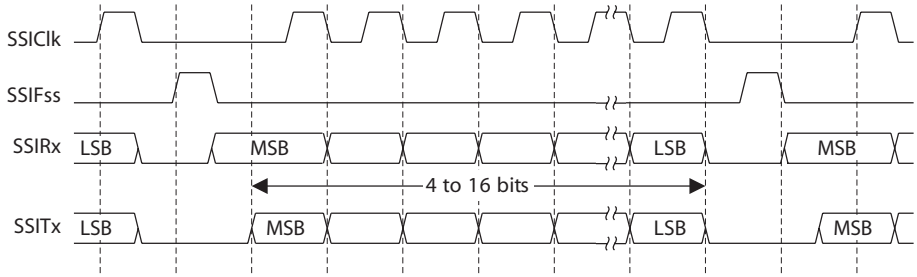


Figure 12-5. Freescale SPI Format (Continuous Transfer) with SPO=0 and SPH=0



In this configuration, during idle periods:

- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. This causes slave data to be enabled onto the SSIRx input line of the master. The master SSITx output pad is enabled.

One half SSIClk period later, valid master data is transferred to the SSITx pin. Now that both the master and slave data have been set, the SSIClk master clock pin goes High after one further half SSIClk period.

The data is now captured on the rising and propagated on the falling edges of the SSIClk signal.

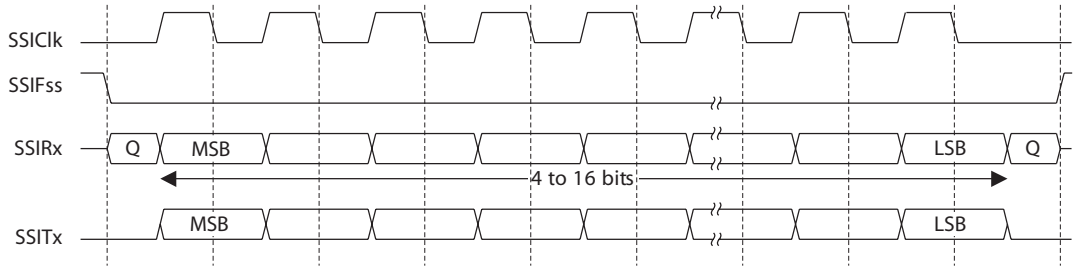
In the case of a single word transmission, after all bits of the data word have been transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSIFss signal must be pulsed High between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is logic zero. Therefore, the master device must raise the SSIFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSIFss pin is returned to its idle state one SSIClk period after the last bit has been captured.

12.2.4.4 Freescale SPI Frame Format with SPO=0 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=0 and SPH=1 is shown in Figure 12-6, which covers both single and continuous transfers.

Figure 12-6. Freescale SPI Frame Format with SPO=0 and SPH=1



In this configuration, during idle periods:

- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. The master SSITx output is enabled. After a further one half SSIClk period, both master and slave valid data is enabled onto their respective transmission lines. At the same time, the SSIClk is enabled with a rising edge transition.

Data is then captured on the falling edges and propagated on the rising edges of the SSIClk signal.

In the case of a single word transfer, after all bits have been transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

For continuous back-to-back transfers, the SSIFss pin is held Low between successive data words and termination is the same as that of the single word transfer.

12.2.4.5 Freescale SPI Frame Format with SPO=1 and SPH=0

Single and continuous transmission signal sequences for Freescale SPI format with SPO=1 and SPH=0 are shown in Figure 12-7 and Figure 12-8.

Figure 12-7. Freescale SPI Frame Format (Single Transfer) with SPO=1 and SPH=0

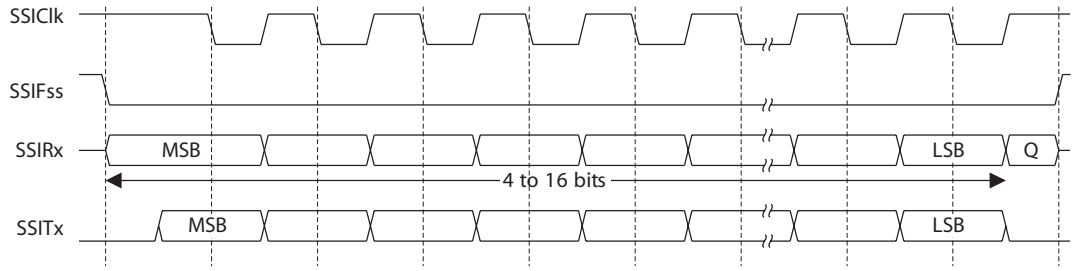
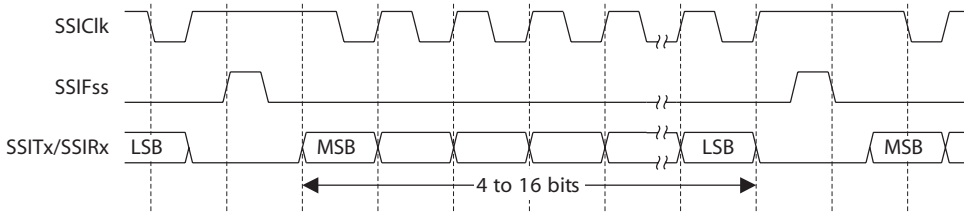


Figure 12-8. Freescale SPI Frame Format (Continuous Transfer) with SPO=1 and SPH=0



In this configuration, during idle periods:

- SSIClk is forced High
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low, which causes slave data to be immediately transferred onto the SSIRx line of the master. The master SSITx output pad is enabled.

One half period later, valid master data is transferred to the SSITx line. Now that both the master and slave data have been set, the SSIClk master clock pin becomes Low after one further half SSIClk period. This means that data is captured on the falling edges and propagated on the rising edges of the SSIClk signal.

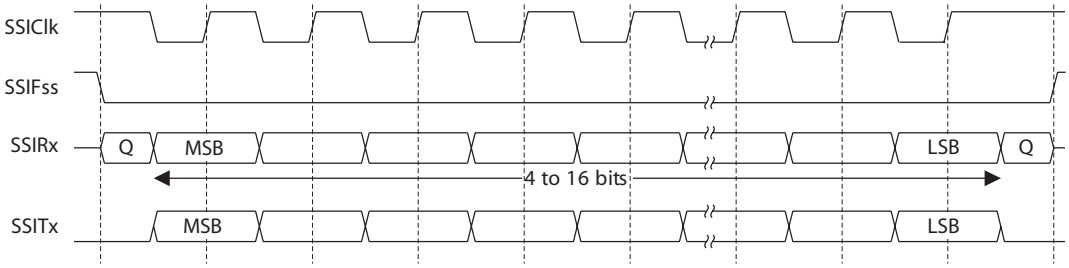
In the case of a single word transmission, after all bits of the data word are transferred, the SSIFss line is returned to its idle High state one SSIClk period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSIFss signal must be pulsed High between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the SPH bit is logic zero. Therefore, the master device must raise the SSIFss pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSIFss pin is returned to its idle state one SSIClk period after the last bit has been captured.

12.2.4.6 Freescale SPI Frame Format with SPO=1 and SPH=1

The transfer signal sequence for Freescale SPI format with SPO=1 and SPH=1 is shown in Figure 12-9, which covers both single and continuous transfers.

Figure 12-9. Freescale SPI Frame Format with SPO=1 and SPH=1



Note: Q is undefined in Figure 12-9.

In this configuration, during idle periods:

- SSIClk is forced High
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low
- When the SSI is configured as a master, it enables the SSIClk pad
- When the SSI is configured as a slave, it disables the SSIClk pad

If the SSI is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSIFss master signal being driven Low. The master SSITx output pad is enabled. After a further one-half SSIClk period, both master and slave data are enabled onto their respective transmission lines. At the same time, SSIClk is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SSIClk signal.

After all bits have been transferred, in the case of a single word transmission, the SSIFss line is returned to its idle high state one SSIClk period after the last bit has been captured.

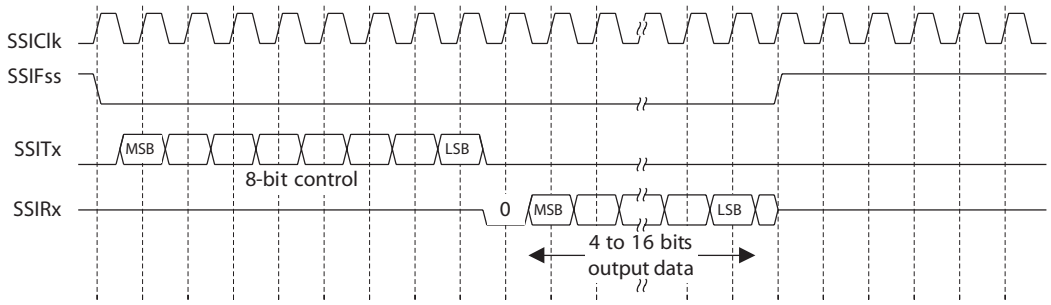
For continuous back-to-back transmissions, the SSIFss pin remains in its active Low state, until the final bit of the last word has been captured, and then returns to its idle state as described above.

For continuous back-to-back transfers, the SSIFss pin is held Low between successive data words and termination is the same as that of the single word transfer.

12.2.4.7 National Semiconductor MICROWIRE Frame Format

Figure 12-10 shows the National Semiconductor MICROWIRE frame format, again for a single frame. Figure 12-11 shows the same format when back-to-back frames are transmitted.

Figure 12-10. National Semiconductor MICROWIRE Frame Format (Single Frame)



MICROWIRE format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex, using a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSI to the off-chip slave device. During this transmission, no incoming data is received by the SSI. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bits in length, making the total frame length anywhere from 13 to 25 bits.

In this configuration, during idle periods:

- SSIClk is forced Low
- SSIFss is forced High
- The transmit data line SSITx is arbitrarily forced Low

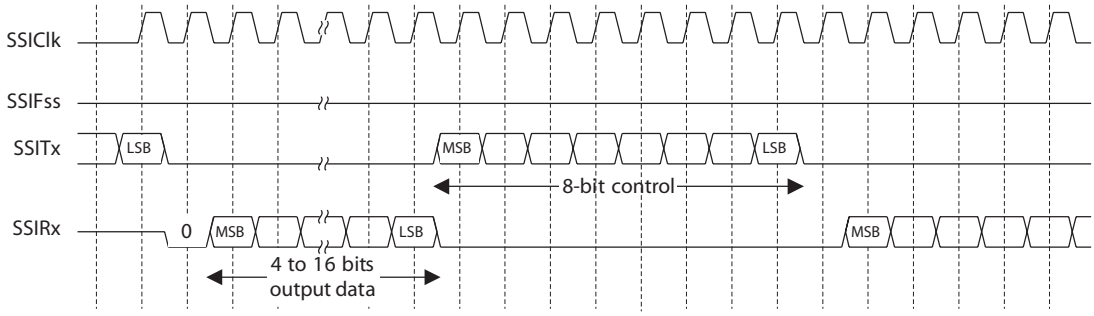
A transmission is triggered by writing a control byte to the transmit FIFO. The falling edge of SSIFss causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic, and the MSB of the 8-bit control frame to be shifted out onto the SSITx pin. SSIFss remains Low for the duration of the frame transmission. The SSIRx pin remains tristated during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on the rising edge of each SSIClk. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state, and the slave responds by transmitting data back to the SSI. Each bit is driven onto the SSIRx line on the falling edge of SSIClk. The SSI in turn latches each bit on the rising edge of SSIClk. At the end of the frame, for single transfers, the SSIFss signal is pulled High one clock period after the last bit has been latched in the receive serial shifter, which causes the data to be transferred to the receive FIFO.

Note: The off-chip slave device can tristate the receive line either on the falling edge of SSIClk after the LSB has been latched by the receive shifter, or when the SSIFss pin goes High.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the SSIFss line is continuously asserted (held Low) and transmission of data occurs back-to-back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge of SSIClk, after the LSB of the frame has been latched into the SSI.

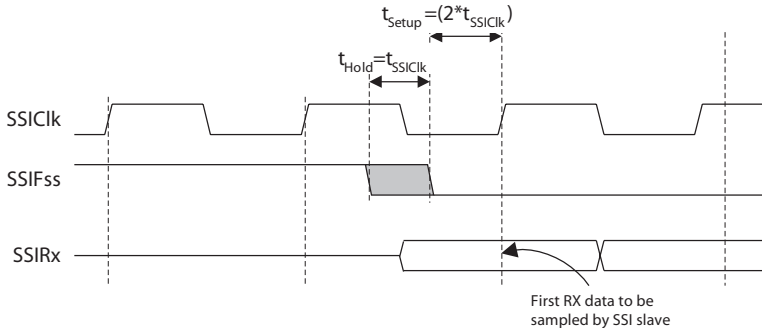
Figure 12-11. National Semiconductor MICROWIRE Frame Format (Continuous Transfers)



In the MICROWIRE mode, the SSI slave samples the first bit of receive data on the rising edge of SSIClk after SSIFss has gone Low. Masters that drive a free-running SSIClk must ensure that the SSIFss signal has sufficient setup and hold margins with respect to the rising edge of SSIClk.

Figure 12-12 illustrates these setup and hold time requirements. With respect to the SSIClk rising edge on which the first bit of receive data is to be sampled by the SSI slave, SSIFss must have a setup of at least two times the period of SSIClk on which the SSI operates. With respect to the SSIClk rising edge previous to this edge, SSIFss must have a hold of at least one SSIClk period.

Figure 12-12. National Semiconductor MICROWIRE Frame Format, SSIFss Input Setup and Hold Requirements



12.3 Initialization and Configuration

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 will be a master or slave:
   a. For master operations, set the **SSICR1** register to 0x00000000.
   b. For slave mode (output enabled), set the **SSICR1** register to 0x00000004.
   c. For slave mode (output disabled), set the **SSICR1** register to 0x0000000C.
3. Configure the clock prescale divisor by writing the **SSICPSR** register.
4. Write the **SSICR0** register with the following configuration:
   a. Serial clock rate (SCR)
   b. Desired clock phase/polarity, if using Freescale SPI mode (SPH/SPO)
   c. The protocol mode: Freescale SPI, TI SSF, National Semiconductor MICROWIRE (FRF)
   d. The data size (DSS)
5. Enable the SSI by setting the SSE bit in the **SSICR1** register.

As an example, assume the SSI must be configured to operate with the following parameters:

- Master operation
- Freescale SPI mode (SPO=1, SPH=1)
- 1 Mbps bit rate
- 8 data bits

Assuming the system clock is 20 MHz, the bit rate calculation would be:

```
FSSIClk = FSysClk / (CPSDVR * (1 + SCR)) ' 1x106 = 20x106 / (CPSDVR * (1 + SCR))
```

In this case, if CPSDVR=2, SCR must be 9.

The configuration sequence would be as follows:

1. Ensure that the SSE bit in the **SSICR1** register is disabled.
2. Write the **SSICR1** register with a value of 0x00000000.
3. Write the **SSICPSR** register with a value of 0x00000002.

4. Write the **SSICR0** register with a value of 0x000009C7.
5. The SSI is then enabled by setting the SSE bit in the **SSICR1** register to 1.

12.4 Register Map

Table 12-1 lists the SSI registers. All addresses given are relative to the SSI base address of 0x40008000.

Note: The SSI must be disabled (see the SSE bit in the **SSICR1** register) before any of the control registers are reprogrammed.

Table 12-1. SSI Register Map

Offset	Name	Reset	Type	Description	See page
0x000	SSICR0	0x00000000	RW	Control 0	229
0x004	SSICR1	0x00000000	RW	Control 1	231
0x008	SSIDR	0x00000000	RW	Data	232
0x00C	SSISR	0x00000003	RO	Status	233
0x010	SSICPSR	0x00000000	RW	Clock prescale	234
0x014	SSIIM	0x00000000	RW	Interrupt mask	235
0x018	SSIRIS	0x00000008	RO	Raw interrupt status	236
0x01C	SSIMIS	0x00000000	RO	Masked interrupt status	237
0x020	SSIICR	0x00000000	W1C	Interrupt clear	238
0xFD0	SSIPeriphID4	0x00000000	RO	Peripheral identification 4	239
0xFD4	SSIPeriphID5	0x00000000	RO	Peripheral identification 5	240
0xFD8	SSIPeriphID6	0x00000000	RO	Peripheral identification 6	241
0xFDC	SSIPeriphID7	0x00000000	RO	Peripheral identification 7	242
0xFE0	SSIPeriphID0	0x00000022	RO	Peripheral identification 0	243
0xFE4	SSIPeriphID1	0x00000000	RO	Peripheral identification 1	244
0xFE8	SSIPeriphID2	0x00000018	RO	Peripheral identification 2	245
0xFEC	SSIPeriphID3	0x00000001	RO	Peripheral identification 3	246
0xFF0	SSIPCellID0	0x0000000D	RO	PrimeCell identification 0	247
0xFF4	SSIPCellID1	0x000000F0	RO	PrimeCell identification 1	248
0xFF8	SSIPCellID2	0x00000005	RO	PrimeCell identification 2	249
0xFFC	SSIPCellID3	0x000000B1	RO	PrimeCell identification 3	250

12.5 Register Descriptions

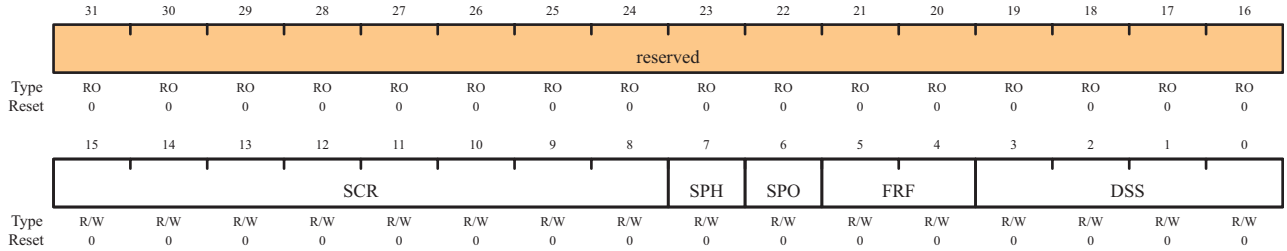
The remainder of this section lists and describes the SSI registers, in numerical order by address offset.

Register 1: SSI Control 0 (SSICR0), offset 0x000

SSICR0 is control register 0 and contains bit fields that control various functions within the SSI module. Functionality such as protocol mode, clock rate and data size are configured in this register.

SSI Control 0 (SSICR0)

Offset 0x000



Bit	Name	Type	Reset	Description
31:16	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
15:8	SCR	R/W	0	SSI Serial Clock Rate The value SCR is used to generate the transmit and receive bit rate of the SSI. The bit rate is: BR= F_{SSICLK}/(CPSDVR * (1 + SCR)) where CPSDVR 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 SSIClk pin. If SPO is 1, a steady state High value is placed on the SSIClk pin when data is not being transferred.

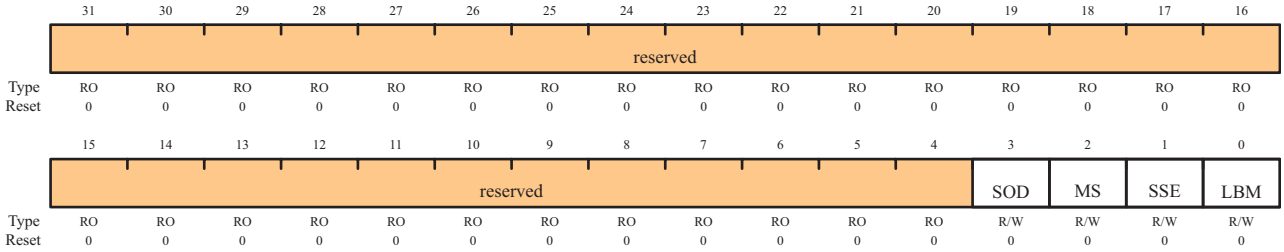
Bit	Name	Type	Reset	Description
5:4	FRF	R/W	0	SSI Frame Format Select. The FRF values are defined as follows: FRF Value: Frame Format 00: Freescale SPI Frame Format 01: Texas Instruments Synchronous Serial Frame Format 10: National Semiconductor MICROWIRE Frame Format
3:0	DSS	R/W	0	SSI Data Size Select The DSS values are defined as follows: DSS Value: Data Size 0000-0010: Reserved 0011: 4-bit data 0100: 5-bit data 0101: 6-bit data 0110: 7-bit data 0111: 8-bit data 1000: 9-bit data 1001: 10-bit data 1010: 11-bit data 1011: 12-bit data 1100: 13-bit data 1101: 14-bit data 1110: 15-bit data 1111: 16-bit data

Register 2: SSI Control 1 (SSICR1), offset 0x004

SSICR1 is control register 1 and contains bit fields that control various functions within the SSI module. Master and slave mode functionality is controlled by this register.

SSI Control 1 (SSCR1)

Offset 0x004



Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
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 will not drive the SSITx pin. 0: SSI can drive SSITx output in Slave Output mode. 1: SSI must not drive the SSITx output in Slave mode.
2	MS	R/W	0	SSI Master/Slave Select This bit selects Master or Slave mode and can be modified only when SSI is disabled (SSE=0). 0: Device configured as a master. 1: Device configured as a slave.
1	SSE	R/W	0	SSI Synchronous Serial Port Enable Setting this bit enables SSI operation. **Note:** This bit must be set to 0 before any control registers are reprogrammed. 0: SSI operation disabled. 1: SSI operation enabled.
0	LBM	R/W	0	SSI Loopback Mode Setting this bit enables Loopback Test mode. 0: Normal serial port operation enabled. 1: Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.

Register 3: SSI Data (SSIDR), offset 0x008

SSIDR is the data register and is 16-bits wide. When **SSIDR** is read, the entry in the receive FIFO (pointed to by the current FIFO read pointer) is accessed. As data values are removed by the SSI receive logic from the incoming data frame, they are placed into the entry in the receive FIFO (pointed to by the current FIFO write pointer).

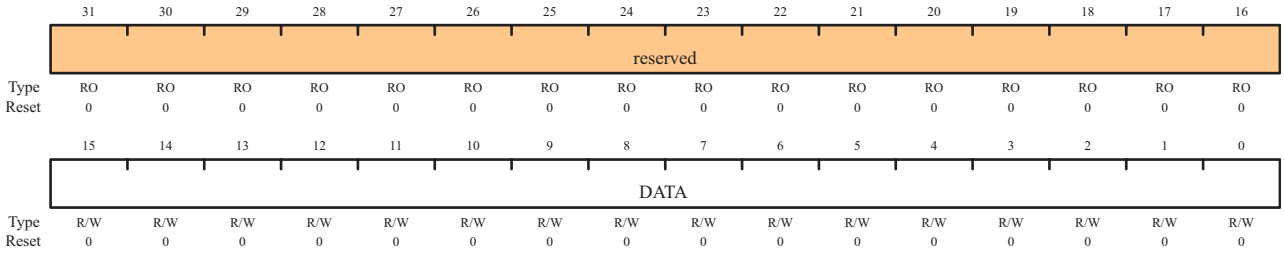
When **SSIDR** is written to, the entry in the transmit FIFO (pointed to by the write pointer), is written to. Data values are removed from the transmit FIFO one value at a time by the transmit logic. It is loaded into the transmit serial shifter, then serially shifted out onto the SSITx pin at the programmed bit rate.

When a data size of less than 16 bits is selected, the user must right-justify data written to the transmit FIFO. The transmit logic ignores the unused bits. Received data less than 16 bits is automatically right-justified in the receive buffer.

When the SSI is programmed for National Semiconductor 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)

Offset 0x008



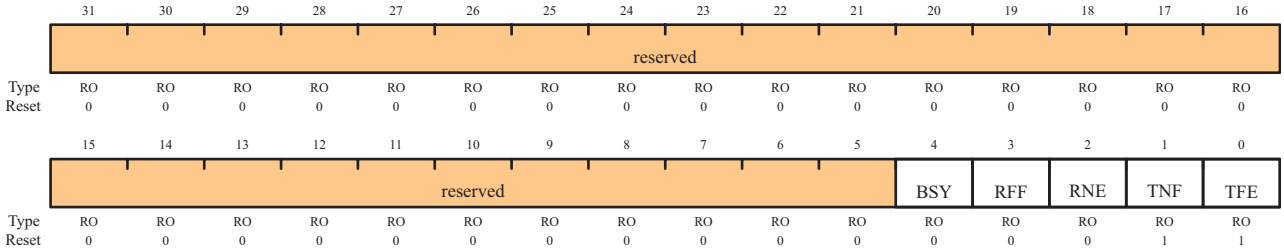
Bit/Field	Name	Type	Reset	Description
31:16	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
15:0	DATA	R/W	0	SSI Receive/Transmit Data A read operation reads the receive FIFO. A write operation writes the transmit FIFO. Software must right-justify data when the SSI is programmed for a data size that is less than 16 bits. Unused bits at the top are ignored by the transmit logic. The receive logic automatically right-justifies the data.

Register 4: SSI Status (SSISR), offset 0x00C

SSISR is a status register that contains bits that indicate the FIFO fill status and the SSI busy status.

SSI Status (SSISR)

Offset 0x00C



Bit/Field	Name	Type	Reset	Description
31:5	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
4	BSY	RO	0	SSI Busy Bit 0: SSI is idle. 1: SSI is currently transmitting and/or receiving a frame, or the transmit FIFO is not empty.
3	RFF	RO	0	SSI Receive FIFO Full 0: Receive FIFO is not full. 1: Receive FIFO is full.
2	RNE	RO	0	SSI Receive FIFO Not Empty 0: Receive FIFO is empty. 1: Receive FIFO is not empty.
1	TNF	RO	1	SSI Transmit FIFO Not Full 0: Transmit FIFO is full. 1: Transmit FIFO is not full.
0	TFE	R0	1	SSI Transmit FIFO Empty 0: Transmit FIFO is not empty. 1: Transmit FIFO is empty.

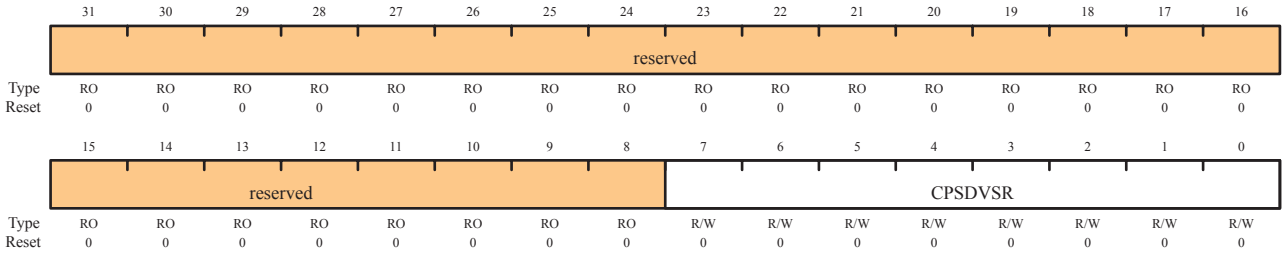
Register 5: SSI Clock Prescale (SSICPSR), offset 0x010

SSICPSR is the clock prescale register and specifies the division factor by which the system clock must be internally divided before further use.

The value programmed into this register must be an even number between 2 and 254. The least-significant bit of the programmed number is hard-coded to zero. If an odd number is written to this register, data read back from this register has the least-significant bit as zero.

SSI Clock Prescale (SSICPSR)

Offset 0x010



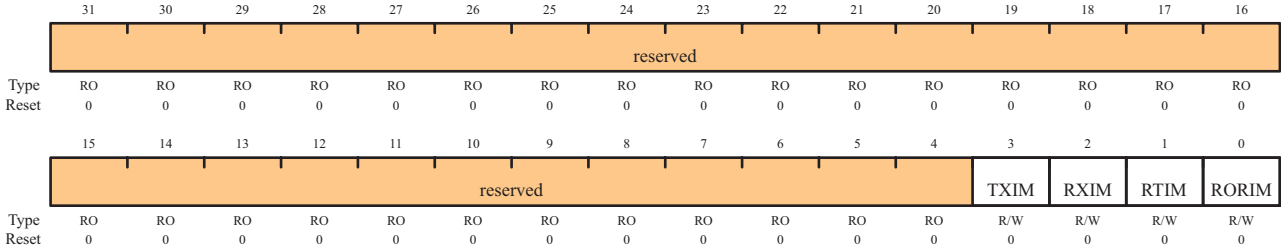
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CPSDVSR	R/W	0	SSI Clock Prescale Divisor This value must be an even number from 2 to 254, depending on the frequency of `SSIClk`. The LSB always returns 0 on reads.

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)
Offset 0x014

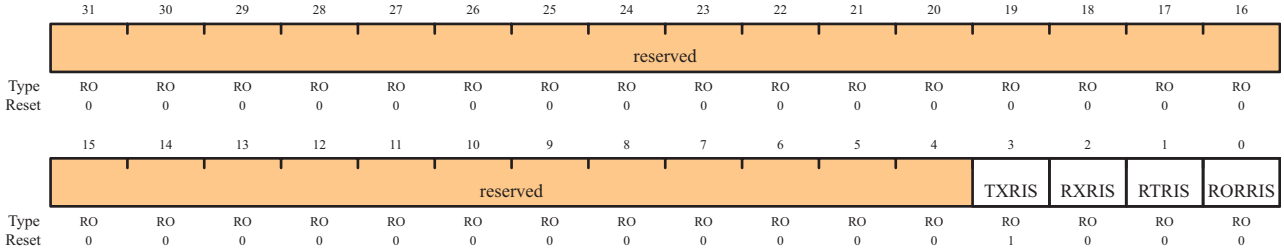


Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
3	TXIM	R/W	0	SSI Transmit FIFO Interrupt Mask 0: TX FIFO half-empty or less condition interrupt is masked. 1: TX FIFO half-empty or less condition interrupt is not masked.
2	RXIM	R/W	0	SSI Receive FIFO Interrupt Mask 0: RX FIFO half-full or less condition interrupt is masked. 1: RX FIFO half-full or less condition interrupt is not masked.
1	RTIM	R/W	0	SSI Receive Time-Out Interrupt Mask 0: RX FIFO time-out interrupt is masked. 1: RX FIFO time-out interrupt is not masked.
0	RORIM	R/W	0	SSI Receive Overrun Interrupt Mask 0: RX FIFO overrun interrupt is masked. 1: RX FIFO overrun interrupt is not masked.

Register 7: SSI Raw Interrupt Status (SSIRIS), offset 0x018

The **SSIRIS** register is the raw interrupt status register. On a read, this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

SSI Raw Interrupt Status (SSIRIS)
Offset 0x018

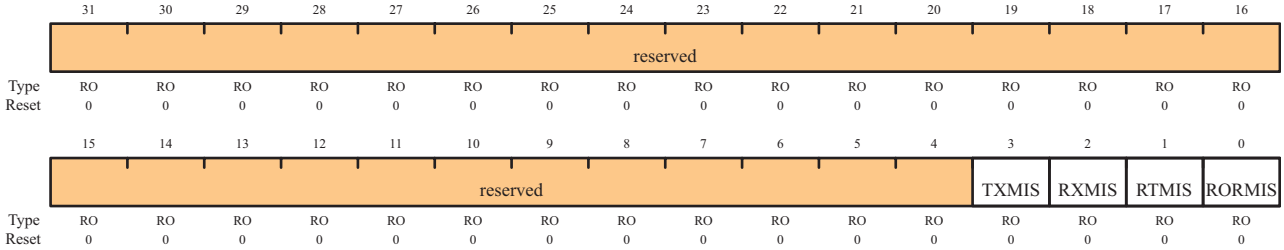


Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
3	TXRIS	RO	1	SSI Transmit FIFO Raw Interrupt Status Indicates that the transmit FIFO is half empty or more, when set.
2	RXRIS	RO	0	SSI Receive FIFO Raw Interrupt Status Indicates that the receive FIFO is half empty 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)
Offset 0x01C



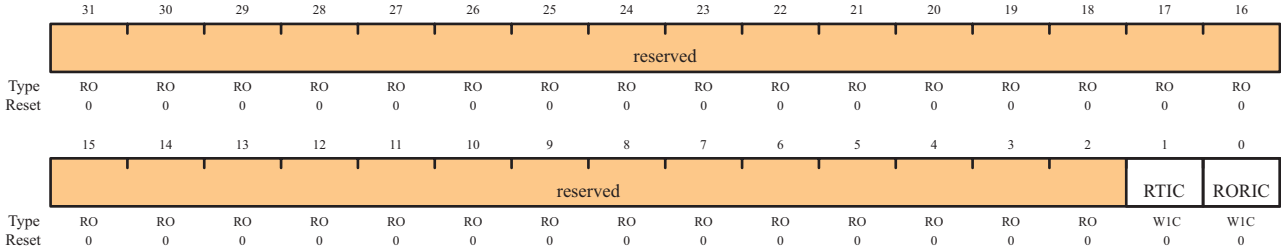
Bit/Field	Name	Type	Reset	Description
31:4	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
3	TXMIS	RO	0	SSI Transmit FIFO Masked Interrupt Status Indicates that the transmit FIFO is half empty or more, when set.
2	RXMIS	RO	0	SSI Receive FIFO Masked Interrupt Status Indicates that the receive FIFO is half empty 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 **SSIPICR** 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)

Offset 0x020



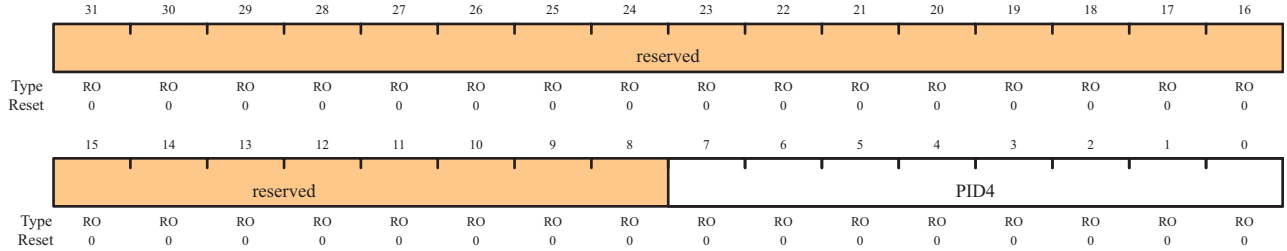
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
1	RTIC	W1C	0	SSI Receive Time-Out Interrupt Clear 0: No effect on interrupt. 1: Clears interrupt.
0	RORIC	W1C	0	SSI Receive Overrun Interrupt Clear 0: No effect on interrupt. 1: Clears interrupt.

Register 10: SSI Peripheral Identification 4 (SSIPeriphID4), offset 0xFD0

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 4 (SSIPeriphID4)

Offset 0xFD0

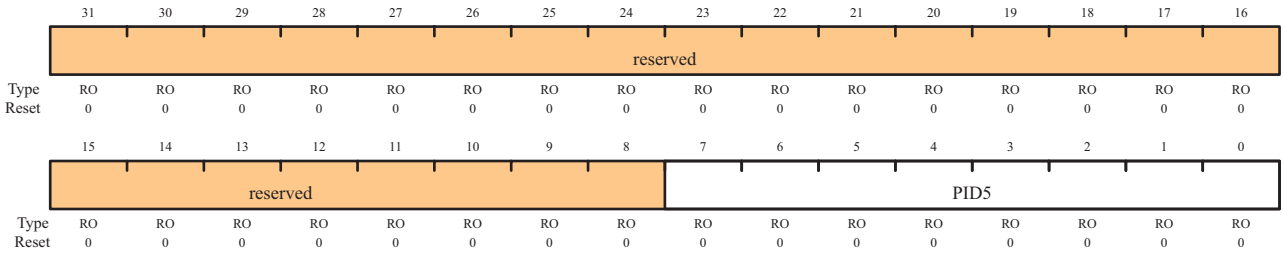


Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID4	RO	0x00	SSI Peripheral ID Register[7:0]

Register 11: SSI Peripheral Identification 5 (SSIPeriphID5), offset 0xFD4

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 5 (SSIPeriphID5)
Offset 0xFD4



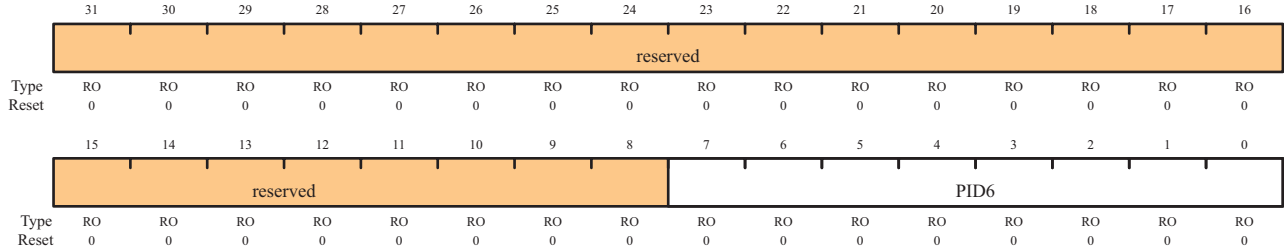
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID5	RO	0x00	SSI Peripheral ID Register[15:8]

Register 12: SSI Peripheral Identification 6 (SSIPeriphID6), offset 0xFD8

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 6 (SSIPeriphID6)

Offset 0xFD8



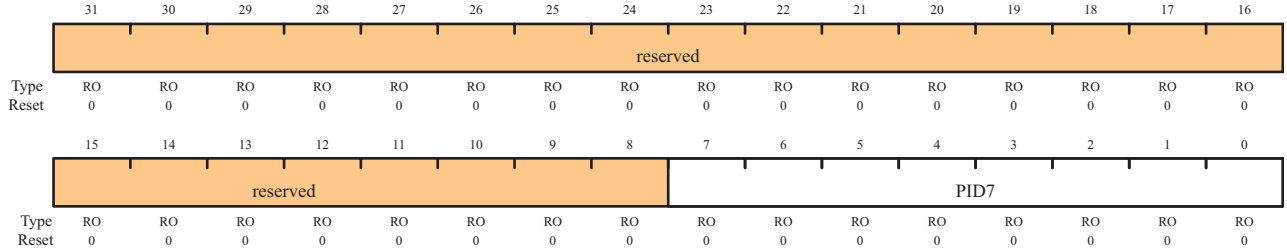
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID6	RO	0x00	SSI Peripheral ID Register[23:16]

Register 13: SSI Peripheral Identification 7 (SSIPeriphID7), offset 0xFDC

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 7 (SSIPeriphID7)

Offset 0xFDC



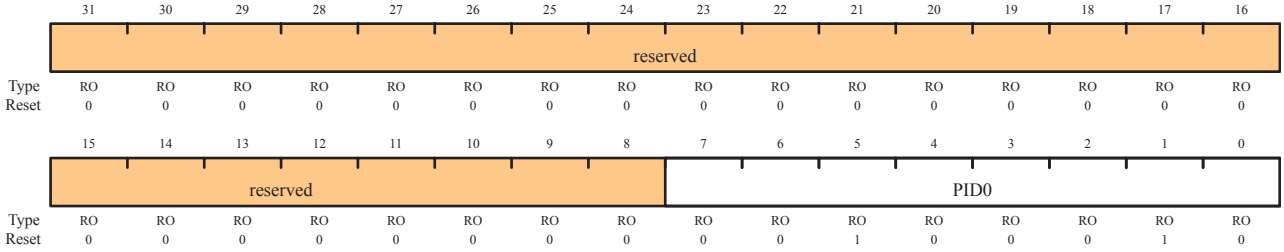
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID7	RO	0x00	SSI Peripheral ID Register[31:24]

Register 14: SSI Peripheral Identification 0 (SSIPeriphID0), offset 0xFE0

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 0 (SSIPeriphID0)

Offset 0xFEO



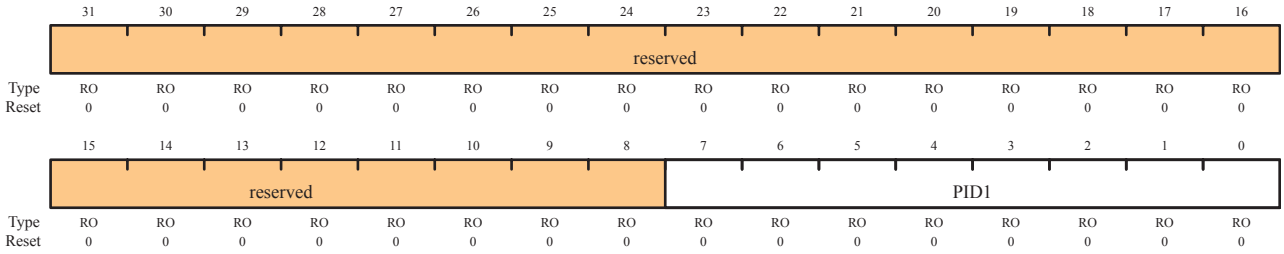
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID0	RO	0x22	SSI Peripheral ID Register[7:0] Can be used by software to identify the presence of this peripheral.

Register 15: SSI Peripheral Identification 1 (SSIPeriphID1), offset 0xFE4

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 1 (SSIPeriphID1)

Offset 0xFE4



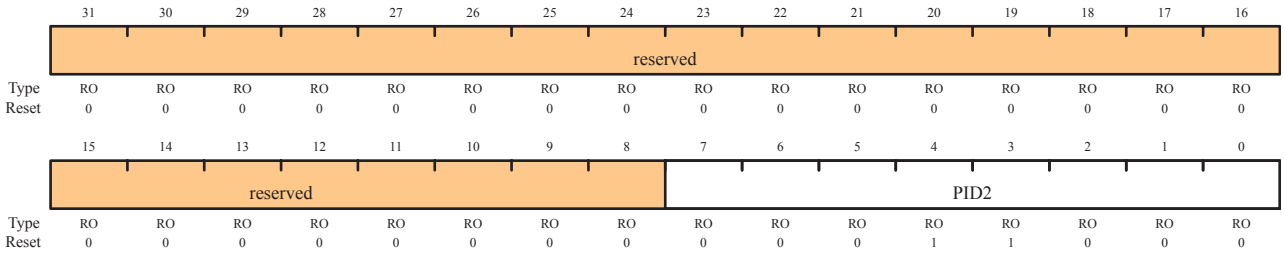
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID1	RO	0x0	SSI Peripheral ID Register [15:8] Can be used by software to identify the presence of this peripheral.

Register 16: SSI Peripheral Identification 2 (SSIPeriphID2), offset 0xFE8

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 2 (SSIPeriphID2)

Offset 0xFE8



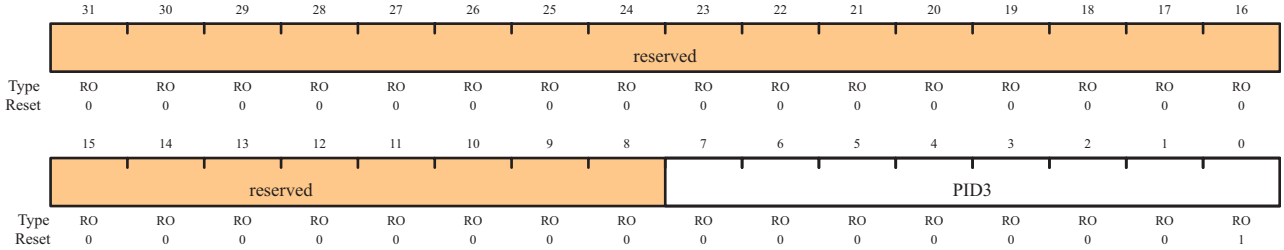
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID2	RO	0x18	SSI Peripheral ID Register [23:16] Can be used by software to identify the presence of this peripheral.

Register 17: SSI Peripheral Identification 3 (SSIPeriphID3), offset 0xFEC

The **SSIPeriphIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Peripheral Identification 3 (SSIPeriphID3)

Offset 0xFEC



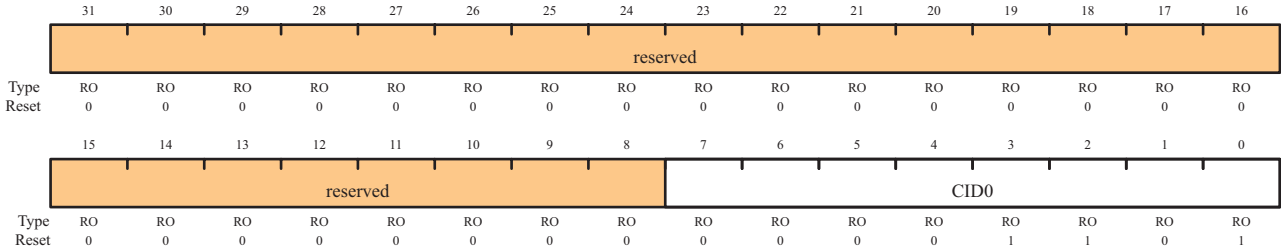
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	PID3	RO	0x1	SSI Peripheral ID Register [31:24] Can be used by software to identify the presence of this peripheral.

Register 18: SSI PrimeCell Identification 0 (SSIPCellID0), offset 0xFF0

The **SSIPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Primecell Identification 0 (SSIPCellID0)

Offset 0xFF0



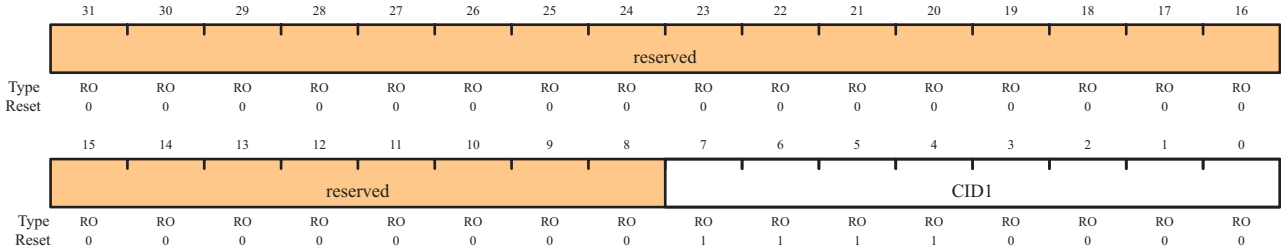
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CID0	RO	0x0D	SSI PrimeCell ID Register [7:0] Provides software a standard cross-peripheral identification system.

Register 19: SSI PrimeCell Identification 1 (SSIPCellID1), offset 0xFF4

The **SSIPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Primecell Identification 1 (SSIPCellID1)

Offset 0xFF4



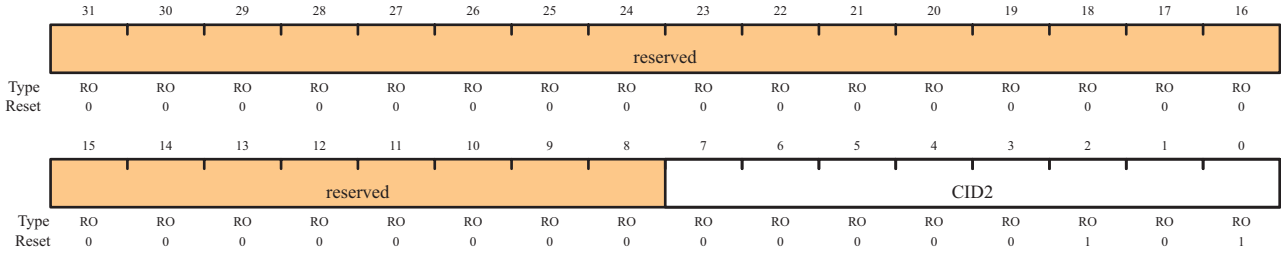
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CID1	RO	0xF0	SSI PrimeCell ID Register [15:8] Provides software a standard cross-peripheral identification system.

Register 20: SSI PrimeCell Identification 2 (SSIPCellID2), offset 0xFF8

The **SSIPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Primecell Identification 2 (SSIPCellID2)

Offset 0xFF8



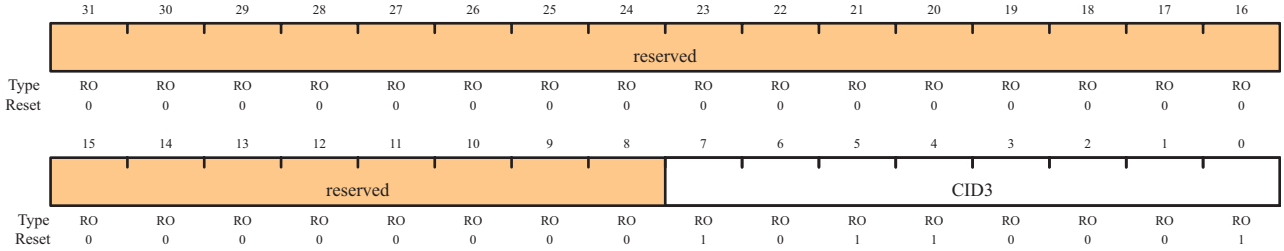
Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CID2	RO	0x05	SSI PrimeCell ID Register [23:16] Provides software a standard cross-peripheral identification system.

Register 21: SSI PrimeCell Identification 3 (SSIPCellID3), offset 0xFFC

The **SSIPCellIDn** registers are hard-coded and the fields within the register determine the reset value.

SSI Primecell Identification 3 (SSIPCellID3)

Offset 0xFFC



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
7:0	CID3	RO	0xB1	SSI PrimeCell ID Register [31:24] Provides software a standard cross-peripheral identification system.

13 Analog Comparators

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

The LM3S101 controller provides two independent integrated analog comparators that can be configured to drive an output or generate an interrupt.

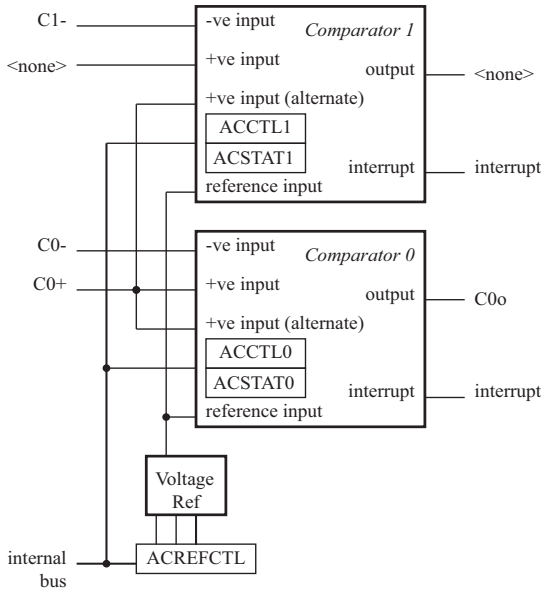
A comparator can compare a test voltage against any one of these voltages:

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

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

13.1 Block Diagram

Figure 13-1. Analog Comparator Block Diagram



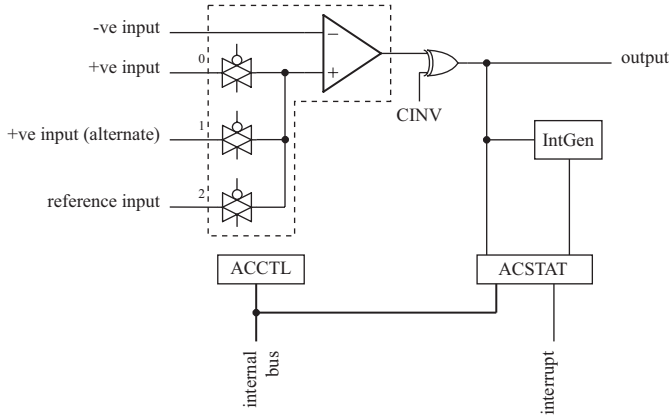
13.2 Functional Description

Important: If a comparator input is configured as an analog input that is not a full scale value (0 V or 3.3 V only), the input Schmitt Trigger is required to be disabled via the GPIO module.

The comparator compares the VIN- and VIN+ inputs to produce an output, VOUT.

As shown in Figure 13-2, 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 13-2. Structure of Comparator Unit



A comparator is configured through two status/control registers (**ACCTL** and **ACSTAT**). The internal reference is configured through one control register (**ACREFCTL**). Interrupt status and control is configured through three registers (**ACMIS**, **ACRIS**, and **ACINTEN**). The operating modes of the comparators are shown in Table 13-1 and Table 13-2.

Typically, the comparator output is used internally to generate controller interrupts. It may also be used to drive an external pin.

Note: The proper pad configuration for the comparator input and output pins are described in Table 8-1 on page 97.

Table 13-1. Comparator 0 Operating Modes

ACCNTL0	Comparator 0			
ASRCP	**VIN-**	**VIN+**	**Output**	**Interrupt**
00	C0-	C0+	C0o/C1-	yes
01	C0-	C0+	C0o/C1-	yes
10	C0-	Vref	C0o/C1-	yes
11	C0-	reserved	C0o/C1-	yes

Table 13-2. Comparator 1 Operating Modes

ACCNTL1	Comparator 1			
ASRCP	**VIN-**	**VIN+**	**Output**	**Interrupt**
00	C0o/C1-[a]	n/a	n/a	yes
01	C0o/C1-	C0+	n/a	yes
10	C0o/C1-	Vref	n/a	yes
11	C0o/C1-	reserved	n/a	yes

a. C0o and C1- signals share a single pin and may only be used as one or the other.

13.2.1 Internal Reference Programming

The structure of the internal reference is shown in Figure 13-3. This is controlled by a single configuration register (**ACREFCTL**). Table 13-3 shows the programming options to develop specific internal reference values, to compare an external voltage against a particular voltage generated internally.

Figure 13-3. Comparator Internal Reference Structure

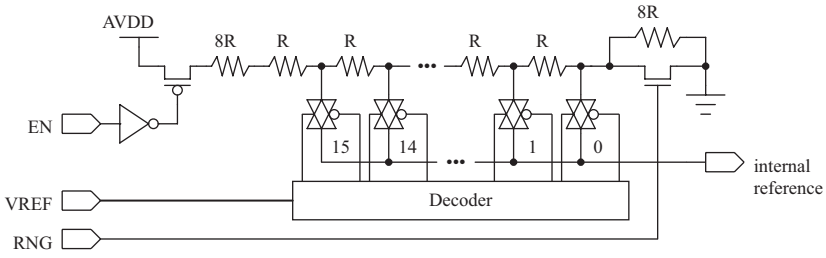


Table 13-3. Internal Reference Voltage and ACREFCTL Field Values

ACREFCTL Register		Output Reference Voltage Based on VREF Field Value
EN Bit Value	**RNG Bit Value**	
EN=0	RNG=X	0 V (GND) for any value of VREF; however, it is recommended that RNG=1 and VREF=0 for the least noisy ground reference.
EN=1	RNG=0	Total resistance in ladder is 32 R. $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF + 8)}{32}$ $V_{REF} = 0.825 + 0.103 \cdot VREF$ The range of internal reference in this mode is 0.825–2.37 V.
	RNG=1	Total resistance in ladder is 24 R. $V_{REF} = AV_{DD} \times \frac{R_{VREF}}{R_T}$ $V_{REF} = AV_{DD} \times \frac{(VREF)}{24}$ $V_{REF} = 0.1375 \cdot VREF$ The range of internal reference for this mode is 0.0–2.0625 V.

13.3 Register Map

Table 13-4 lists the comparator registers. All addresses given are relative to the Analog Comparator base address of 0x4003C000.

Table 13-4. Analog Comparator Register Map

Offset	Name	Reset	Type	Description	See page
0x00	ACMIS	0x00000000	RO	Interrupt status	255
0X04	ACRIS	0x00000000	RO	Raw interrupt status	256
0X08	ACINTEN	0x00000000	R/W	Interrupt enable	257
0x10	ACREFCTL	0x00000000	R/W	Reference voltage control	258
0x20	ACSTAT0	0x00000000	RO	Comparator 0 status	259
0x40	ACSTAT1	0x00000000	RO	Comparator 1 status	259
0x24	ACCTL0	0x00000000	RW	Comparator 0 control	260
0x44	ACCTL1	0x00000000	RW	Comparator 1 control	260

13.4 Register Descriptions

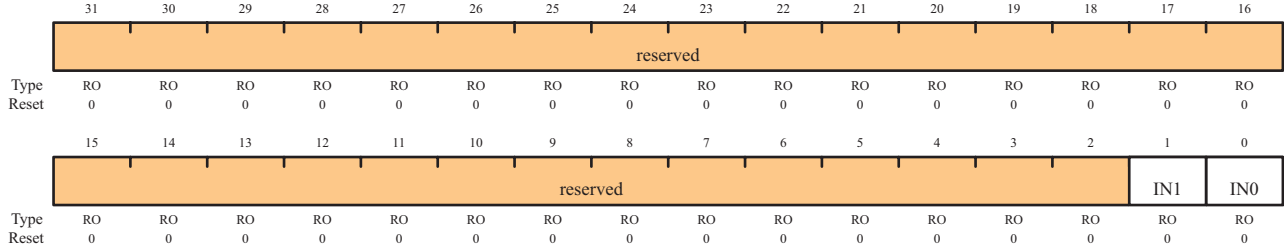
The remainder of this section lists and describes the Analog Comparator registers, in numerical order by address offset.

Register 1: Analog Comparator Masked Interrupt Status (ACMIS), offset 0x00

This register provides a summary of the interrupt status (masked) of the comparators.

Analog Comparator Masked Interrupt Status (ACMIS)

Offset 0x000



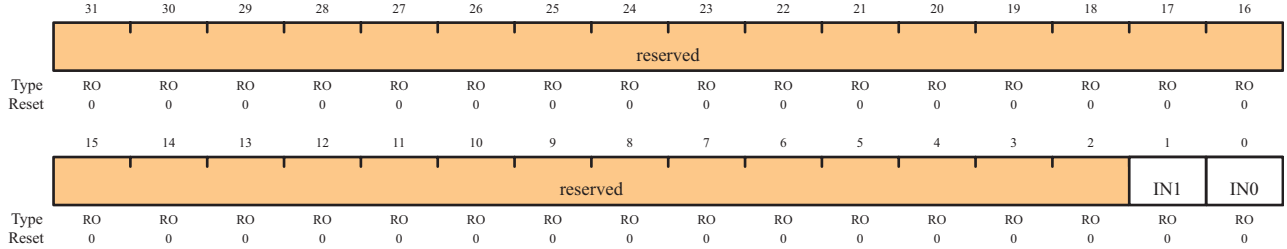
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
1	IN1	RO	0	Comparator 1 Masked Interrupt Status Gives the masked interrupt state of this interrupt.
0	IN0	RO	0	Comparator 0 Masked Interrupt Status Gives the masked interrupt state of this interrupt.

Register 2: Analog Comparator Raw Interrupt Status (ACRIS), offset 0x04

This register provides a summary of the interrupt status (raw) of the comparators.

Analog Comparator Raw Interrupt Status (ACRIS)

Offset 0x04



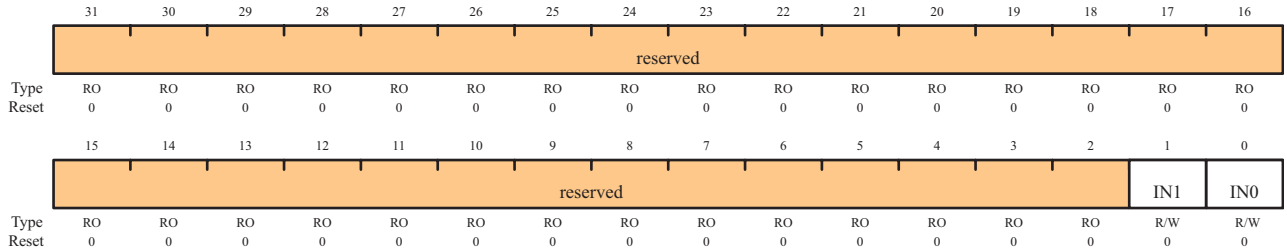
Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
1	IN1	RO	0	When set, indicates that an interrupt has been generated by comparator 1.
0	IN0	RO	0	When set, indicates that an interrupt has been generated by comparator 0.

Register 3: Analog Comparator Interrupt Enable (ACINTEN), offset 0x08

This register provides the interrupt enable for the comparators.

Analog Comparator Interrupt Enable (ACINTEN)

Offset 0x08

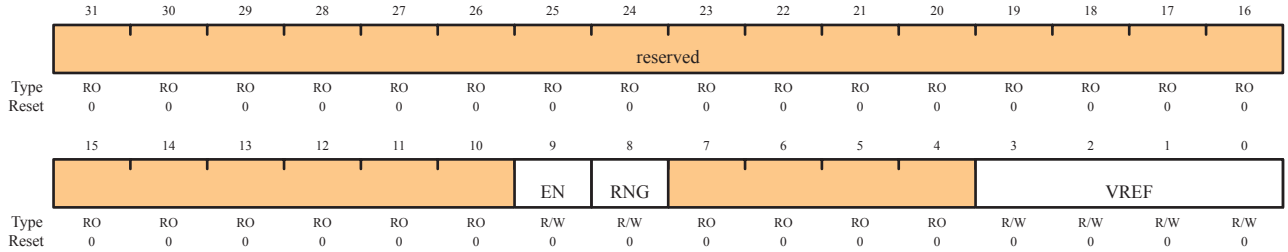


Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
1	IN1	R/W	0	When set, enables the controller interrupt from the comparator 1 output.
0	IN0	R/W	0	When set, enables the controller interrupt from the comparator 0 output.

Register 4: Analog Comparator Reference Voltage Control (ACREFCTL), offset 0x10

This register specifies whether the resistor ladder is powered on as well as the range and tap.

Analog Comparator Reference Voltage Control (ACREFCTL)
Offset 0x010



Bit/Field	Name	Type	Reset	Description
31:10	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
9	EN	R/W	0	The EN bit specifies whether the resistor ladder is powered on. If 0, the resistor ladder is unpowered. If 1, the resistor ladder is connected to the analog V_{DD}. This bit is reset to 0 so that the internal reference consumes the least amount of power if not used and programmed.
8	RNG	R/W	0	The RNG bit specifies the range of the resistor ladder. If 0, the resistor ladder has a total resistance of 32 R. If 1, the resistor ladder has a total resistance of 24 R.
7:4	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
3:0	VREF	R/W	0	The VREF bit field specifies the resistor ladder tap that is passed through an analog multiplexer. The voltage corresponding to the tap position is the internal reference voltage available for comparison.

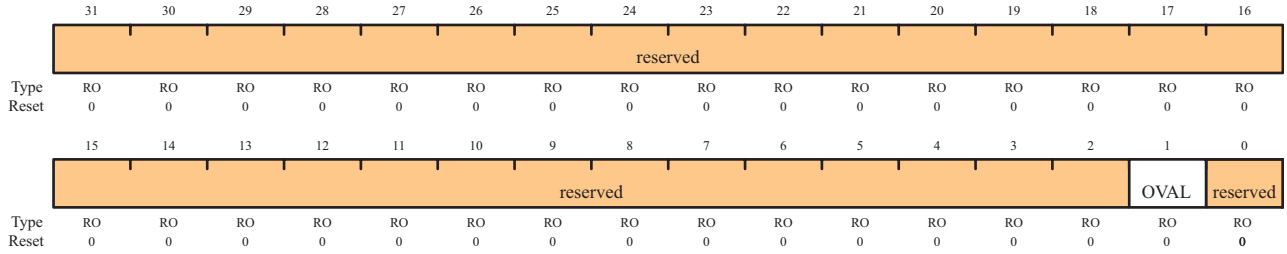
Register 5: Analog Comparator Status 0 (ACSTAT0), offset 0x20

Register 6: Analog Comparator Status 1 (ACSTAT1), offset 0x40

These registers specify the current output value of that comparator.

Analog Comparator Status 0 (ACSTAT0)

Offset 0x020



Bit/Field	Name	Type	Reset	Description
31:2	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
1	OVAL	RO	0	The OVAL bit specifies the current output value of the comparator.
0	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.

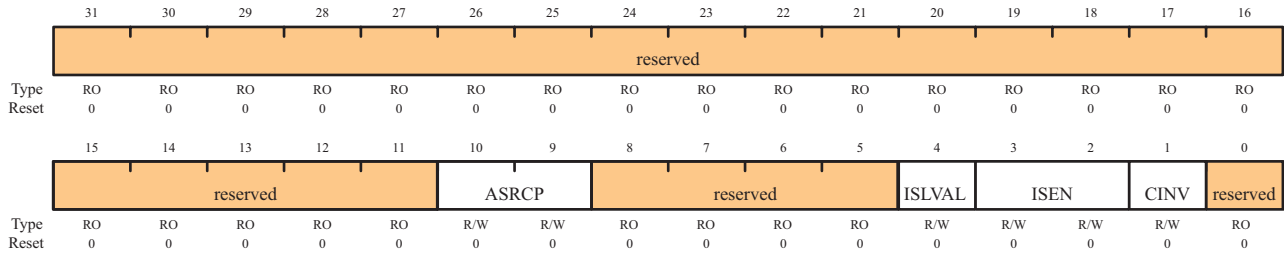
Register 7: Analog Comparator Control 0 (ACCTL0), offset 0x24

Register 8: Analog Comparator Control 1 (ACCTL1), offset 0x44

These registers configure that comparator's input and output.

Analog Comparator Control 0 (ACCTL0)

Offset 0x024



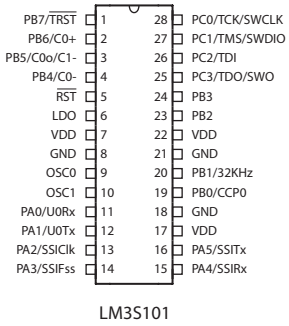
Bit/Field	Name	Type	Reset	Description
31:11	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
10:9	ASRCP	R/W	0	The ASRCP field specifies the source of input voltage to the VIN+ terminal of the comparator. The encodings for this field are as follows: ASRCP Function 00 Pin value 01 Pin value of C0+ 10 Internal voltage reference 11 Reserved
8:5	reserved	RO	0	Reserved bits return an indeterminate value, and should never be changed.
4	ISLVAL	R/W	0	The ISLVAL bit specifies the sense value of the input that generates an interrupt if in Level Sense mode. If 0, an interrupt is generated if the comparator output is Low. Otherwise, an interrupt is generated if the comparator output is High.

Bit/Field	Name	Type	Reset	Description
3:2	ISEN	R/W	0	The `ISEN` field specifies the sense of the comparator output that generates an interrupt. The sense conditioning is as follows: ISEN — Function 00 — Level sense, see ISLVAL 01 — Falling edge 10 — Rising edge 11 — Either edge
1	CINV	R/W	0	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	Reserved bits return an indeterminate value, and should never be changed.

14 Pin Diagram

Figure 14-1 shows the pin diagram and pin-to-signal-name mapping.

Figure 14-1. Pin Connection Diagram



15 Signal Tables

The following tables list the signals available for each pin. Functionality is enabled by software with the **GPIOAFSEL** register (see page 109).

Important: All multiplexed pins are GPIOs by default, with the exception of the five JTAG pins (PB7 and PC[3:0]) which default to the JTAG functionality.

Table 15-1 shows the pin-to-signal-name mapping, including functional characteristics of the signals. Table 15-2 lists the signals in alphabetical order by signal name. Table 15-3 groups the signals by functionality. Table 15-4 lists the GPIO pins and their alternate functionality.

Table 15-1. Signals by Pin Number (Sheet 1 of 2)

Pin Number	Signal Name	Pin Type	Buffer Type	Description
1	PB7	I/O	TTL	GPIO port B bit 7.
	$\overline{\text{TRST}}$	I	TTL	JTAG TAP reset input.
2	PB6	I/O	TTL	GPIO port B bit 6.
	C0+	I	Analog	Analog comparator 0 positive reference input.
3	PB5	I/O	TTL	GPIO port B bit 5.
	C0o	O	TTL	Analog comparator 0 output.
	C1–	I	Analog	Analog comparator 1 negative reference input.
4	PB4	I/O	TTL	GPIO port B bit 4.
	C0–	I	Analog	Analog comparator 0 negative reference input.
5	$\overline{\text{RST}}$	I	TTL	System reset input.
6	LDO	-	Power	The linear drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater.
7	VDD	-	Power	Positive supply for logic and I/O pins.
8	GND	-	Power	Ground reference for logic and I/O pins.
9	OSC0	I	Analog	Oscillator crystal input or an external clock reference input.
10	OSC1	O	Analog	Oscillator crystal output.
11	PA0	I/O	TTL	GPIO port A bit 0.
	U0Rx	I	TTL	UART0 receive data input.
12	PA1	I/O	TTL	GPIO port A bit 1.
	U0Tx	O	TTL	UART0 transmit data output.
13	PA2	I/O	TTL	GPIO port A bit 2.
	SSIClk	I/O	TTL	SSI clock reference (input when in slave mode and output in master mode).

Table 15-1. Signals by Pin Number (Sheet 2 of 2)

Pin Number	Signal Name	Pin Type	Buffer Type	Description
14	PA3	I/O	TTL	GPIO port A bit 3.
	SSIFss	I/O	TTL	SSI frame enable (input for an SSI slave device and output for an SSI master device).
15	PA4	I/O	TTL	GPIO port A bit 4.
	SSIRx	I	TTL	SSI receive data input.
16	PA5	I/O	TTL	GPIO port A bit 5.
	SSITx	O	TTL	SSI transmit data output.
17	VDD	-	Power	Positive supply for logic and I/O pins.
18	GND	-	Power	Ground reference for logic and I/O pins.
19	PB0	I/O	TTL	GPIO port B bit 0.
	CCP0	I/O	TTL	Timer 0 capture input, compare output, or PWM output port 0.
20	PB1	I/O	TTL	GPIO port B bit 1.
	32KHz	I	TTL	Timer clock reference input for real-time clock operation.
21	GND	-	Power	Ground reference for logic and I/O pins.
22	VDD	-	Power	Positive supply for logic and I/O pins.
23	PB2	I/O	TTL	GPIO port B bit 2.
24	PB3	I/O	TTL	GPIO port B bit 3.
25	PC3	I/O	TTL	GPIO port C bit 3.
	TDO	O	TTL	JTAG scan test output.
	SWO	O	TTL	Serial-wire output.
26	PC2	I/O	TTL	GPIO port C bit 2.
	TDI	I	TTL	JTAG scan data input.
27	PC1	I/O	TTL	GPIO port C bit 1.
	TMS	I	TTL	JTAG mode select input.
	SWDIO	I/O	TTL	Serial-wire debug input/output.
28	PC0	I/O	TTL	GPIO port C bit 0.
	TCK	I	TTL	JTAG scan clock reference input.
	SWCLK	I	TTL	Serial-wire clock reference input.

Table 15-2. Signals by Signal Name (Sheet 1 of 2)

Signal Name	Pin Number	Pin Type	Buffer Type	Description
32KHz	20	I	TTL	Timer clock reference input for real-time clock operation.
C0+	2	I	Analog	Analog comparator 0 positive reference input.
C0–	4	I	Analog	Analog comparator 0 negative reference input.
C0o	3	O	TTL	Analog comparator 0 output.
C1–	3	I	Analog	Analog comparator 1 negative reference input.
CCP0	19	I/O	TTL	Timer 0 capture input, compare output, or PWM output port 0.
GND	8	-	Power	Ground reference for logic and I/O pins.
GND	18	-	Power	Ground reference for logic and I/O pins.
GND	21	-	Power	Ground reference for logic and I/O pins.
LDO	6	-	Power	The linear drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater.
OSC0	9	I	Analog	Oscillator crystal input or an external clock reference input.
OSC1	10	O	Analog	Oscillator crystal output.
PA0	11	I/O	TTL	GPIO port A bit 0.
PA1	12	I/O	TTL	GPIO port A bit 1.
PA2	13	I/O	TTL	GPIO port A bit 2.
PA3	14	I/O	TTL	GPIO port A bit 3.
PA4	15	I/O	TTL	GPIO port A bit 4.
PA5	16	I/O	TTL	GPIO port A bit 5.
PB0	19	I/O	TTL	GPIO port B bit 0.
PB1	20	I/O	TTL	GPIO port B bit 1.
PB2	23	I/O	TTL	GPIO port B bit 2.
PB3	24	I/O	TTL	GPIO port B bit 3.
PB4	4	I/O	TTL	GPIO port B bit 4.
PB5	3	I/O	TTL	GPIO port B bit 5.
PB6	2	I/O	TTL	GPIO port B bit 6.
PB7	1	I/O	TTL	GPIO port B bit 7.
PC0	28	I/O	TTL	GPIO port C bit 0.
PC1	27	I/O	TTL	GPIO port C bit 1.

Table 15-2. Signals by Signal Name (Sheet 2 of 2)

Signal Name	Pin Number	Pin Type	Buffer Type	Description
PC2	26	I/O	TTL	GPIO port C bit 2.
PC3	25	I/O	TTL	GPIO port C bit 3.
$\overline{\text{RST}}$	5	I	TTL	System reset input.
SSIClk	13	I/O	TTL	SSI clock reference (input when in slave mode and output in master mode).
SSIFss	14	I/O	TTL	SSI frame enable (input for an SSI slave device and output for an SSI master device).
SSIRx	15	I	TTL	SSI receive data input.
SSITx	16	O	TTL	SSI transmit data output.
SWCLK	28	I	TTL	Serial-wire clock reference input.
SWDIO	27	I/O	TTL	Serial-wire debug input/output.
SWO	25	O	TTL	Serial-wire output.
TCK	28	I	TTL	JTAG scan clock reference input.
TDI	26	I	TTL	JTAG scan data input.
TDO	25	O	TTL	JTAG scan test output.
TMS	27	I	TTL	JTAG mode select input.
$\overline{\text{TRST}}$	1	I	TTL	JTAG TAP reset input.
U0Rx	11	I	TTL	UART0 receive data input.
U0Tx	12	O	TTL	UART0 transmit data output.
VDD	7	-	Power	Positive supply for logic and I/O pins.
VDD	17	-	Power	Positive supply for logic and I/O pins.
VDD	22	-	Power	Positive supply for logic and I/O pins.

Table 15-3. Signals by Function, Except for GPIO (Sheet 1 of 2)

Function	Signal Name	Pin Number	Pin Type	Buffer Type	Description
Analog Comparator	C0+	2	I	Analog	Analog comparator 0 positive reference input.
	C0–	4	I	Analog	Analog comparator 0 negative reference input.
	C0o	3	O	TTL	Analog comparator 0 output.
	C1–	3	I	Analog	Analog comparator 1 negative reference input.
General-Purpose Timers	32KHz	20	I	TTL	Timer clock reference input for real-time clock operation.
	CCP0	19	I/O	TTL	Timer 0 capture input, compare output, or PWM output port 0.
JTAG/SWD/SWO	SWCLK	28	I	TTL	Serial wire clock reference input.
	SWDIO	27	I/O	TTL	Serial-wire debug input/output.
	SWO	25	O	TTL	Serial-wire output.
	TCK	28	I	TTL	JTAG scan clock reference input.
	TDI	26	I	TTL	JTAG scan data input.
	TDO	25	O	TTL	JTAG scan test output.
	TMS	27	I	TTL	JTAG mode select input.
	$\overline{\text{TRST}}$	1	I	TTL	JTAG TAP reset input.
Power	GND	8	-	Power	Ground reference for logic and I/O pins.
	GND	18	-	Power	Ground reference for logic and I/O pins.
	GND	21	-	Power	Ground reference for logic and I/O pins.
	LDO	6	-	Power	The linear drop-out regulator output voltage. This pin requires an external capacitor between the pin and GND of 1 µF or greater.
	VDD	7	-	Power	Positive supply for logic and I/O pins.
	VDD	17	-	Power	Positive supply for logic and I/O pins.
	VDD	22	-	Power	Positive supply for logic and I/O pins.

Table 15-3. Signals by Function, Except for GPIO (Sheet 2 of 2)

Function	Signal Name	Pin Number	Pin Type	Buffer Type	Description
SSI	SSIClk	13	I/O	TTL	SSI clock reference (input when in slave mode and output in master mode).
	SSIFss	14	I/O	TTL	SSI frame enable (input for an SSI slave device and output for an SSI master device).
	SSIRx	15	I	TTL	SSI receive data input.
	SSITx	16	O	TTL	SSI transmit data output.
System Control & Clocks	OSC0	9	I	Analog	Oscillator crystal input or an external clock reference input.
	OSC1	10	O	Analog	Oscillator crystal output.
	$\overline{\text{RST}}$	5	I	TTL	System reset input.
UART	U0Rx	11	I	TTL	UART0 receive data input.
	U0Tx	12	O	TTL	UART0 transmit data output.

Table 15-4. GPIO Pins and Alternate Functions (Sheet 1 of 2)

GPIO Pin	Pin Number	Multiplexed Function	Multiplexed Function
PA0	11	U0Rx	
PA1	12	U0Tx	
PA2	13	SSIClk	
PA3	14	SSIFss	
PA4	15	SSIRx	
PA5	16	SSITx	
PB0	19	CCP0	
PB1	20	32KHz	
PB2	23		
PB3	24		
PB4	4	C0-	
PB5	3	C0o	C1-
PB6	2	C0+	
PB7	1	$\overline{\text{TRST}}$	
PC0	28	TCK	SWCLK

Table 15-4. GPIO Pins and Alternate Functions (Sheet 2 of 2)

GPIO Pin	Pin Number	Multiplexed Function	Multiplexed Function
PC1	27	TMS	SWDIO
PC2	26	TDI	
PC3	25	TDO	SWO

16 Operating Characteristics

Table 16-1. Temperature Characteristics

Characteristic	Symbol	Value	Unit
Operating temperature range[a]	T_A	0 to +70 for commercial -40 to +85 for industrial	°C

a. Maximum storage temperature is 150°C.

Table 16-2. Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance (junction to ambient)[a]	θ_{JA}	74	°C/W
Average junction temperature[b]	T_J	$T_A + (P_{AVG} \bullet \theta_{JA})$	°C
Maximum junction temperature	T_{JMAX}	TBD	°C

a. Junction to ambient thermal resistance θ_{JA} numbers are determined by a package simulator.
b. Power dissipation is a function of temperature.

17 Electrical Characteristics

17.1 DC Characteristics

17.1.1 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 17-1. Maximum Ratings

Characteristic[a]	Symbol	Value	Unit
Supply voltage range (V_{DD})	V_{DD}	0 to +3.6	V
Input voltage	V_{IN}	-0.3 to 5.5	V
Maximum current for pins, excluding pins operating as GPIOs	I	±100	mA
Maximum current for GPIO pins	I	±100	mA

a. Voltages are measured with respect to GND.

Important: This device contains circuitry to protect the inputs against damage due to high-static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either V_{SS} or V_{DD}).

17.1.2 Recommended DC Operating Conditions

Table 17-2. Recommended DC Operating Conditions

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{DD}	Supply voltage	3	3.3	3.6	V
V_{SS}	Supply ground	0	0	0	V
V_{IH}	High-level input voltage	2		5	V
V_{IL}	Low-level input voltage	-.3		1.3	V
V_{SIH}	High-level input voltage for Schottky inputs	TBD			V
V_{SIL}	Low-level input voltage for Schottky inputs			TBD	V
V_{OH}	High-level output voltage	-	3.3	TBD	V
V_{OL}	Low-level output voltage	TBD	0	-	V

Table 17-2. Recommended DC Operating Conditions

Parameter	Parameter Name	Min	Nom	Max	Unit
I_{OH}[a]	Low-level source current, V_{OH}=TBD				mA
I_{OL}	High-level source current, V_{OH}=TBD				mA

a. Different GPIO drive strengths.

17.1.3 On-Chip Linear Drop-Out (LDO) Regulator Characteristics

Table 17-3. LDO Regulator Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{LDOOUT}	Programmable internal (logic) power supply output value	0.9		3.3	V
	Output voltage accuracy		2%		
t_{PON}	Power-on time			100	µs
t_{ON}	Time on			200	µs
t_{OFF}	Time off			100	µs
V_{STEP}	Step programming incremental voltage		50		mV
C_{LDO}	External filter capacitor size for internal power supply		1		µF

17.1.4 Power Specifications

- V_{DD}=3.3 V
- LDO=2.5
- Temperature=25 °C
- System Clock=20 MHz (with PLL)

Table 17-4. Power Specifications

Parameter	Parameter Name	Min	Nom	Max	Unit
IDD_{run}	Run mode		TBD		mA

Table 17-4. Power Specifications

Parameter	Parameter Name	Min	Nom	Max	Unit
IDD_{sleep}	Sleep mode		TBD		µA
$IDD_{deepsleep}$	Deep-sleep mode		TBD		µA

17.1.5 Power-Up and Low-Voltage (Brown-Out) Detect Characteristics

Table 17-5. Power-Up and Brown-Out Detect Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
V_{TH}[a]	Power-up threshold voltage		2		V
t_{POR}	Power-up assertion time		10		ms
V_{BTH}	Brown-out threshold voltage			3	V
t_{BOR}	Brown-out assertion time		500		µs

a. The internal power-on reset circuit may be used unless the power supply slew rate (SR_{PS}) is less than the following relation. If so, the brown-out detector triggers immediately after the internal reset is released: $SR_{PS} < (V_{BTH}–V_{TH})/ t_{POR}$

17.1.6 Flash Memory Characteristics

Table 17-6. Flash Memory Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
PE_{cyc}	Maximum number of guaranteed program/ erase cycles[a] before failure	10,000			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.

17.2 AC Characteristics

17.2.1 Load Conditions

TBD

17.2.2 Clocks

Table 17-7. Phase Locked Loop (PLL) Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
$f_{ref\_crystal}$	Crystal reference[a]	3.579545	-	8.192	MHz
f_{ref_ext}	External reference[a]	3.579545	-	8.192	MHz
$f_{ref\_crystal\_bypass}$	PLL bypass	1		8	MHz
$f_{ref\_ext\_bypass}$	PLL bypass[b]	0	-	20	MHz
f_{pll}	PLL frequency[b]		200		MHz
T_{READY}	PLL lock time		-	0.5	ms

a. The exact value is determined by the crystal value programmed into the XTAL field of the **Run-Mode Clock Configuration (RCC)** register (see page 70).
b. PLL frequency is automatically calculated by the hardware based on the XTAL field of the **RCC** register.

Table 17-8. Clock Characteristics

Parameter	Parameter Name	Min	Nom	Max	Unit
f_{BOSC}	Boot oscillator frequency	10.5	15	19.5	MHz
f_{MOSC}	Main oscillator frequency	1	-	8	MHz
t_{MOSC_per}	Main oscillator period	1000	-	125	ms
f_{system_clock}	System clock	0	-	20	MHz

17.2.3 Analog Comparator

TBD

17.2.4 Synchronous Serial Interface (SSI)

Table 17-9. SSI Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
S1	t_{clk_per}	SSIClk cycle time	-	t_{SSIClk}	-	ns
S2	t_{clk_high}	SSIClk high time	-	$(t_{SSIClk})/2$	-	ns
S3	t_{clk_low}	SSIClk low time	-	$(t_{SSIClk})/2$	-	ns
S4	t_{clkrf}	SSIClk rise/fall time	TBD	-	TBD	ns
S5	t_{DMd}	Data from master valid delay time	-	-	TBD	ns
S6	t_{DMs}	Data from master setup time	TBD	-	-	ns
S7	t_{DMh}	Data from master hold time	TBD	-	-	ns
S8	t_{DSs}	Data from slave setup time	TBD	-	-	ns
S9	t_{DSh}	Data from slave hold time	TBD	-	-	ns

Figure 17-1. SSI Timing for TI Frame Format (FRF=01), Single Transfer Timing Measurement

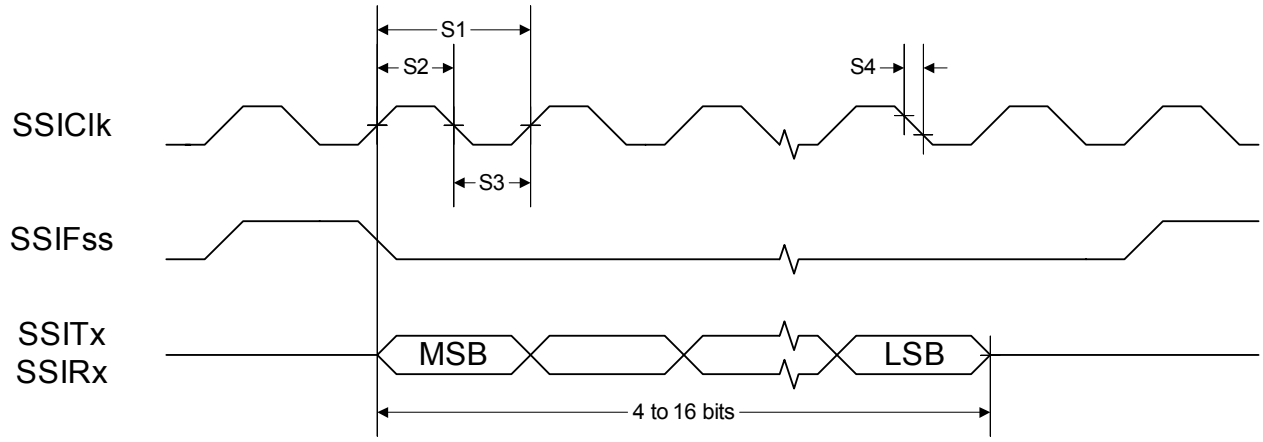


Figure 17-2. SSI Timing for MICROWIRE Frame Format (FRF=10), Single Transfer

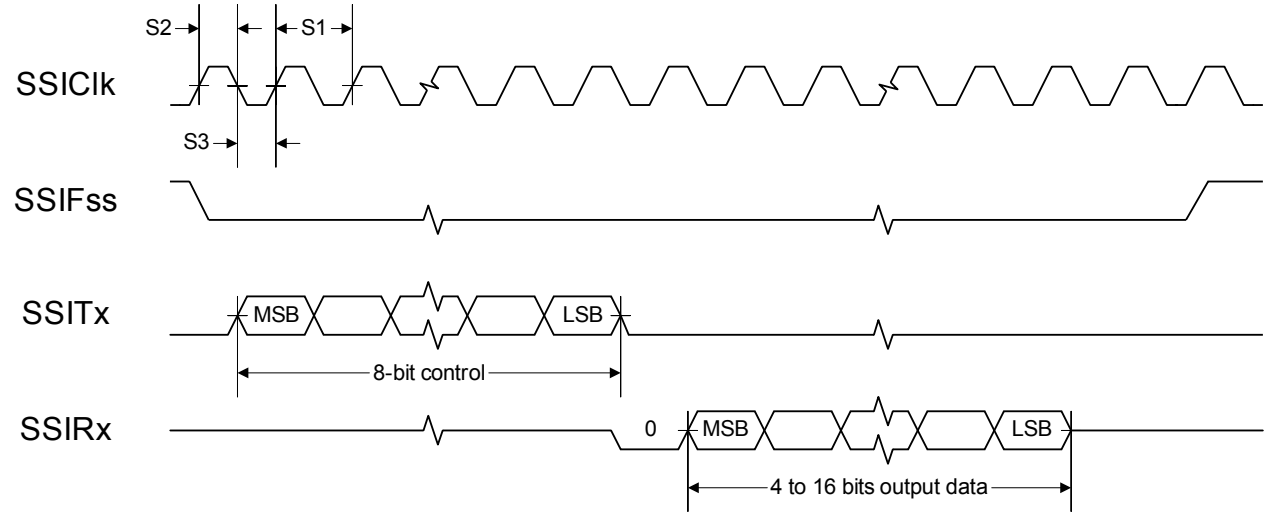
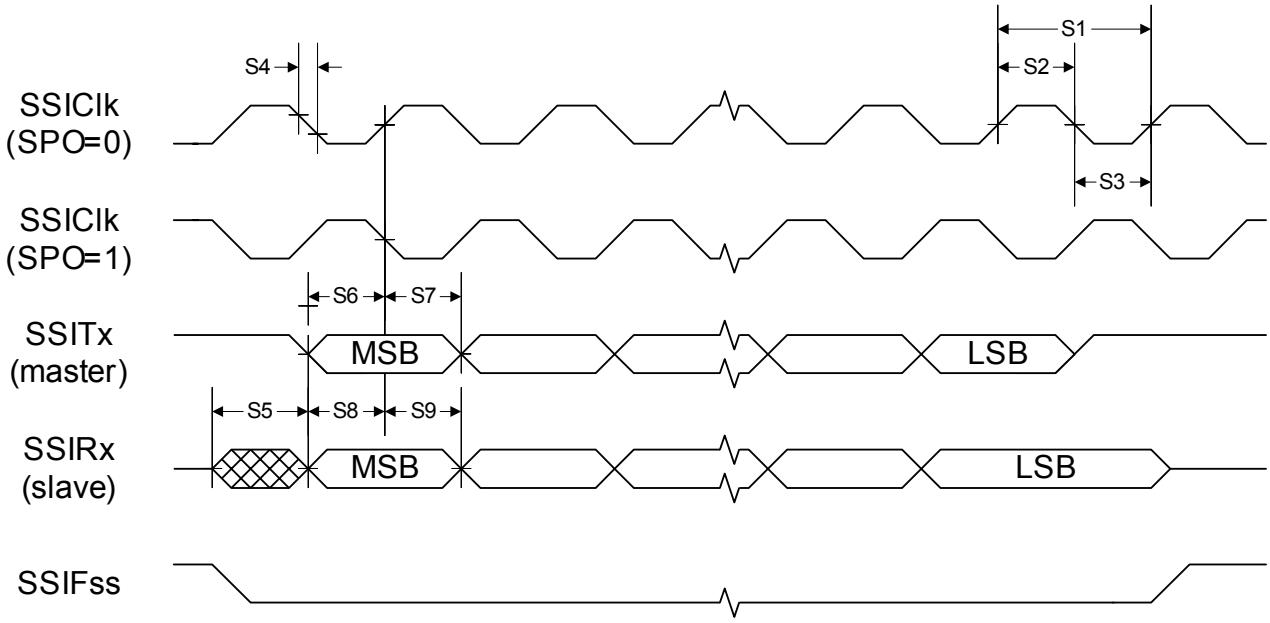


Figure 17-3. SSI Timing for SPI Frame Format (FRF=00), with SPH=1



17.2.5 JTAG and Boundary Scan

Table 17-10. JTAG Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
J1	f_{JCYC}	TCLK Frequency of operation	TBD		TBD	$f_{system\ clock}$
J2	t_{JCYC}	TCLK Cycle Period	TBD		-	t_{CYC}
J3	t_{JCW}	TCLK Pulse Width	TBD		-	ns
J4	t_{JCR}	TCLK Rise time	TBD		TBD	ns
J5	t_{JCF}	TCLK Fall time	TBD		TBD	ns
J6	t_{BSDST}	TDI input data setup time to TCLK Rise	TBD		-	ns
J7	t_{BSDHT}	TDI input data hold time (after TCLK rise)	TBD		-	ns
J8	t_{BSDV}	TCLK Low to Boundary Scan Output Data Valid	TBD		TBD	ns
J9	t_{BSDZ}	TCLK Low to Boundary Scan Output High Z	TBD		TBD	ns
J10	t_{TAPBST}	TMS, TDI Input Data Setup Time to TCLK Rise	TBD		-	ns
J11	t_{TAPBHT}	TMS, TDI Input Data Hold Time after TCLK Rise	TBD		-	ns
J12	t_{TDODV}	TCLK Low to TDO Valid	TBD		TBD	ns
J13	t_{TDODZ}	TCLK Low to TDO High Z	TBD		TBD	ns
J14	t_{TRSTAT}	$\overline{TRST}$ Assert Time	TBD		-	ns
J15	t_{TRSTST}	$\overline{TRST}$ Setup Time (Negation) to TCLK High	TBD		-	ns

Figure 17-4. JTAG Test Clock Input Timing

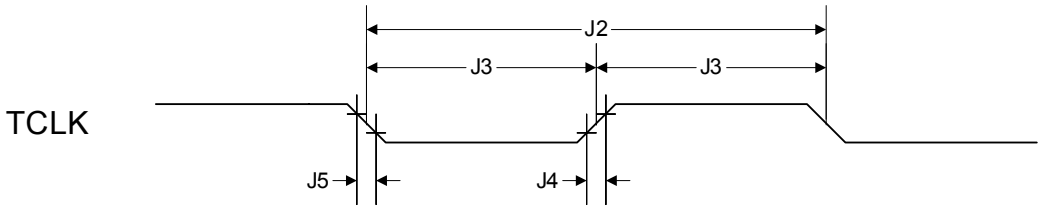


Figure 17-5. JTAG Boundary Scan Timing

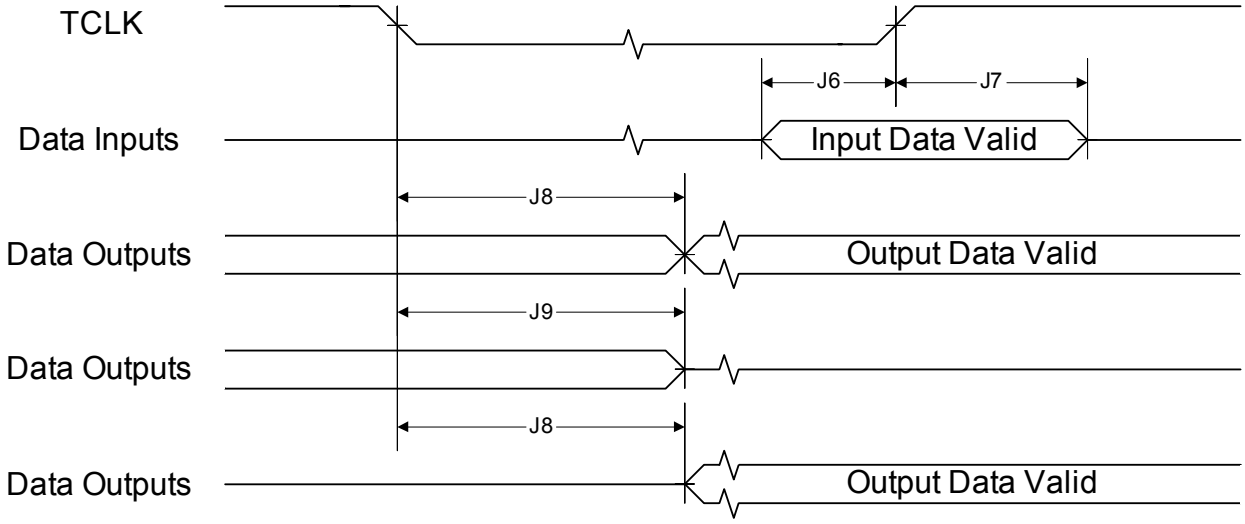


Figure 17-6. JTAG Test Access Port (TAP) Timing

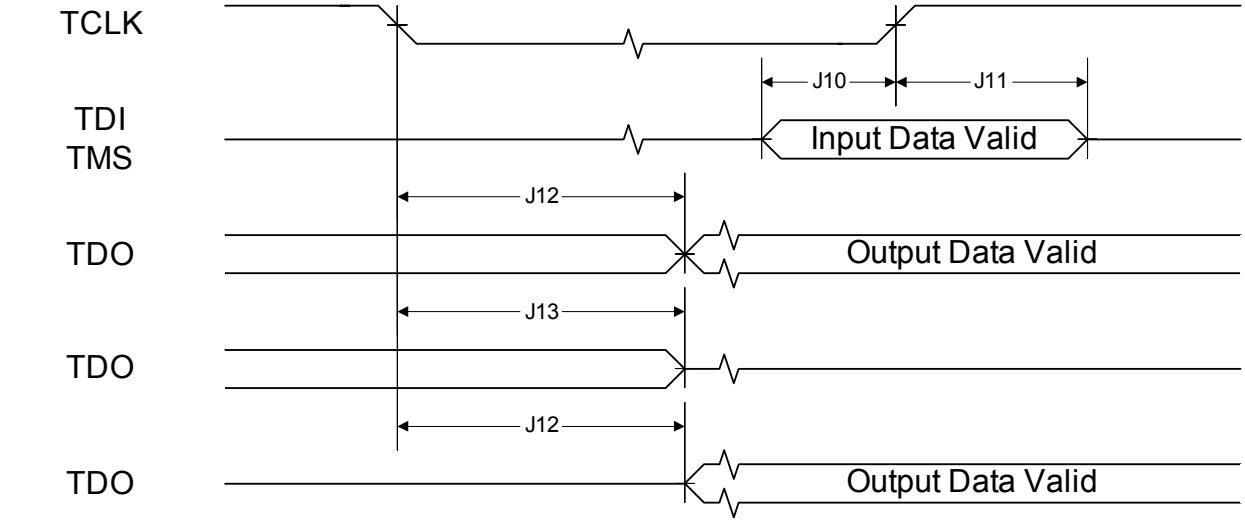
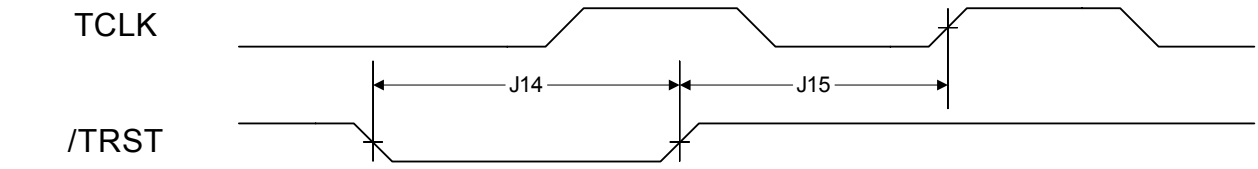


Figure 17-7. JTAG $\overline{\text{TRST}}$ Timing



17.2.6 General-Purpose I/O

Table 17-11. GPIO Characteristics

Parameter	Parameter Name	Condition	Min	Nom	Max	Unit
t_{GPIOR}	GPO Rise Time (output)	2 mA Drive				ns
		4 mA Drive				ns
		8 mA Drive				ns
		8 mA Drive with slew rate control				ns
		OD 2 mA Drive[a]				ns
		OD 4 mA Drive[a]				ns
		OD 8 mA Drive[a]				ns
		OD 8 mA Drive with slew rate control[a]				ns
t_{GPIOF}	GPO Fall Time (output)	2 mA Drive				ns
		4 mA Drive				ns
		8 mA Drive				ns
		8 mA Drive with slew rate control				ns
		OD 2 mA Drive[a]				ns
		OD 4 mA Drive[a]				ns
		OD 8 mA Drive[a]				ns
		OD 8 mA Drive with slew rate control[a]				ns

a. With external 2.8K pull-up resistor. When GPO is in Open Drain (OD) mode, internal pull-up and pull-down resistors are disabled.

17.2.7 Reset

The specifications apply over the full operating temperature range: -55°C to +125°C. Typical values are at ambiant temperature TA=25°C. Test conditions: VIN=3.3 V.

Table 17-12. 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	-		3.0	V
R3	T_{POR}	Power-On Reset Timeout	-	10	-	ms
R4	T_{BOR}	Brown-Out Timeout	-	500	-	µs
R5	T_{IRPOR}	Internal Reset Timeout After POR	15	-	30	ms
R6	T_{IRBOR}	Internal Reset Timeout After BOR[a]	2.5	-	20	µs

Table 17-12. Reset Characteristics

Parameter No.	Parameter	Parameter Name	Min	Nom	Max	Unit
R7	T_{IRHWR}	Internal Reset Timeout After Hardware Reset ($\overline{RST}$ pin)	15	-	30	ms
R8	T_{IRSWR}	Internal Reset Timeout After Software Initiated System Reset[a]	2.5	-	20	µs
R9	T_{IRWDR}	Internal Reset Timeout After Watchdog Reset[a]	2.5	-	20	µs
R10	T_{IRLDOR}	Internal Reset Timeout After LDO Reset[a]	2.5	-	20	µs

a. 20 * t_{MOSC_per}

Figure 17-8. External Reset Timing ($\overline{RST}$)

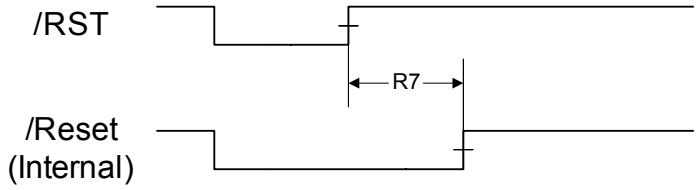


Figure 17-9. Power-On Reset Timing

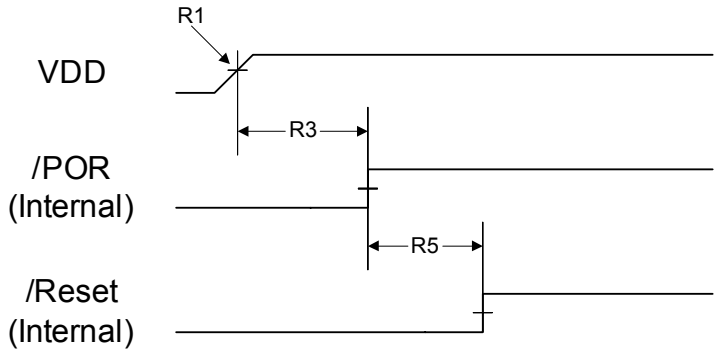


Figure 17-10. Brown-Out Reset Timing

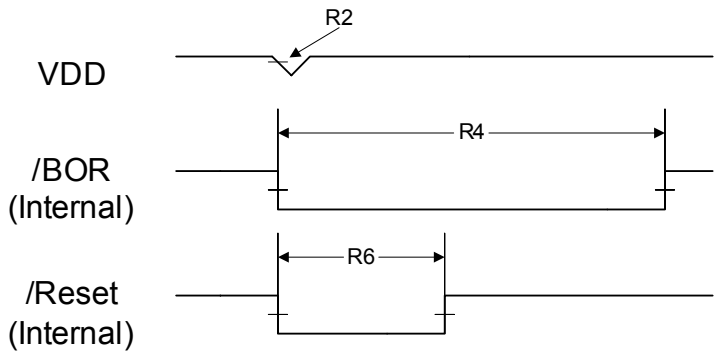


Figure 17-11. Software Reset Timing

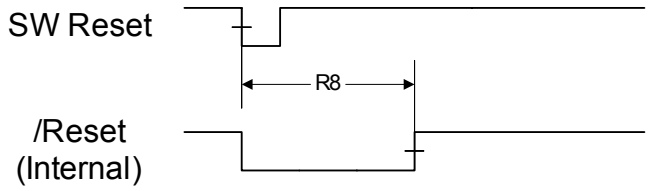


Figure 17-12. Watchdog Reset Timing

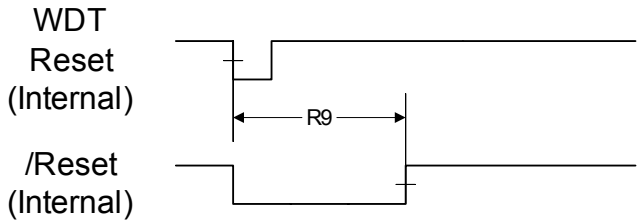
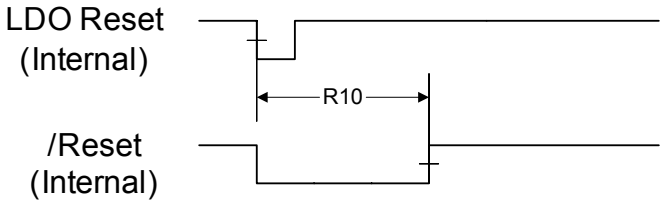
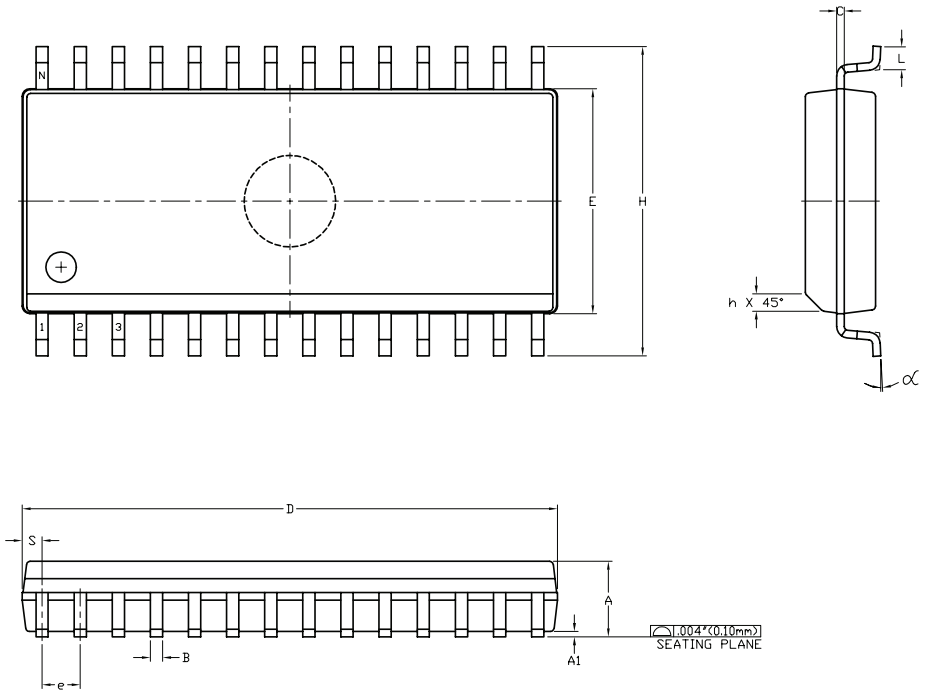


Figure 17-13. LDO Reset Timing



18 Package Information

Figure 18-1. 28-Pin SOIC



NOTES:

1. DIMENSION "D" DOES NOT NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BURRS. MOLD FLASH, PROTRUSIONS AND GATE BURRS SHALL NOT EXCEED .006"(0.15mm) PER SIDE.
2. DIMENSION "E" DOES NOT INCLUDE INTER-LEAD FLASH OR PROTRUSIONS. INTER-LEAD FLASH AND PROTRUSION SHALL NOT EXCEED .010"(0.25mm) PER SIDE.
3. "L" IS THE LENGTH OF TERMINAL FOR SOLDERING TO A SUBSTRATE.
4. "N" IS THE NUMBER OF TERMINAL POSITIONS.
5. TERMINAL NUMBERS ARE SHOWN FOR REFERENCE ONLY.
6. THE LEAD WIDTH "B", AS MEASURED .014"(0.36mm) OR GREATER ABOVE THE SEATING PLANE, SHALL NOT EXCEED A MAXIMUM VALUE OF .024"(0.61mm).
7. REFERENCE DRAWING JEDEC MS013, VARIATION AE.

SYMBOL	DIMENSION IN INCH		DIMENSION IN MM	
	MIN.	MAX.	MIN.	MAX.
A	.093	.104	2.35	2.65
A1	.004	.012	0.10	0.30
B	.013	.020	0.33	0.51
C	.009	.013	0.23	0.32
D	.696	.713	17.70	18.10
E	.291	.299	7.40	7.60
e	.050 BSC		1.27 BSC	
H	.394	.419	10.00	10.65
h	.010	.029	0.25	0.75
L	.016	.050	0.40	1.27
S	.021	.031	0.533	0.787
α	0°	8°	0°	8°

Contact Information

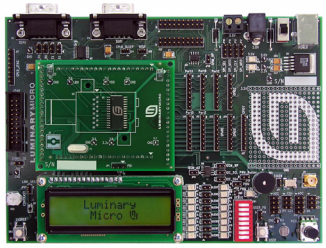
Ordering Information

Part Number	Features													
	Flash (KB)	SRAM (KB)	GPIOs[a]	Timers[b]	UART	SSI	I^2C	Analog Comparators	PWM (CCP Pins)[c]	Operating Temperature[d]	Package[e]	Speed (Clock Frequency in MHz)	Die Revision[f]	Qualification[g]	Shipping Medium[h]
LM3S101-CRN20-XnPT	8	2	2 to 18	2	1	√	-	2	1	C	RN	20	Xn	P	T
LM3S101-CRN20-XnPR															R
LM3S101-CRN20-XnPP															P
LM3S101-IRN20-XnPT										I					T
LM3S101-IRN20-XnPR															R
LM3S101-IRN20-XnPP															P

a. Minimum is number of pins dedicated to GPIO; additional pins are available if certain peripherals are not used. See data sheet for details.
b. One timer available as RTC.
c. PWM motion control functionality can be achieved through the motion control features of the general-purpose timers (using the CCP pins). See data sheet for details.
d. C=Commercial (0 to 70°C); I=Industrial (–40 to 85°C).
e. RN=28-pin RoHS-compliant SOIC.
f. Xn=Part number will contain die revision number at order time, for example, B4.
g. P=Production.
h. T=Tray; R=Rail/Tube; P=Tape and Reel.

Development Kit

The Luminary Micro Stellaris™ Family Development Kit provides the hardware and software tools that engineers need to begin development quickly. Ask your Luminary Micro distributor for part number DK-LM3S101.



Tools to begin development quickly

Company Information

Luminary Micro, Inc. designs, markets, and sells ARM Cortex-M3 based microcontrollers for use in embedded applications within the industrial, commercial, and consumer markets. Luminary Micro is ARM's lead partner in the implementation of the Cortex-M3 core. Please contact us if you are interested in obtaining further information about our company or our products.

Luminary Micro, Inc.
2499 South Capital of Texas Hwy, Suite A-100
Austin, TX 78746
Main: +1-512-279-8800
Fax: +1-512-279-8879
http://www.luminarymicro.com
sales@luminarymicro.com

Support Information

For support on Luminary Micro products, contact:

support@luminarymicro.com
+1-512-279-8800, ext. 3